Chapter 1

Security Principles

In This Chapter

- Why computers aren't secure
- The history of computer security
- The theoretical underpinnings of network security

Security is the sum of all measures taken to prevent loss of any kind. Loss can occur because of user error, defects in code, malicious acts, hardware failure, and acts of nature. With holistic computer security, a number of methods are used to prevent these events, but it's primarily focused on preventing user error and malicious acts.

Security is the antithesis of convenience—generally, the more secure something is, the less convenient it is. Think about this in the context of your life: think of how easy it would be if you could just walk up and push a button to start your car without worrying about keys—or paying for car insurance. But the risk of theft and accidents makes these two security measures mandatory. Meanwhile, advanced technology like remote key fobs for cars is making automotive security easier, just as biometric scanners can make logging on to computers both more secure and less annoying at the same time.

Computer security is not complicated. It may seem that way, but the theory behind computer security is relatively simple. Hacking methods fall into just a few categories. And solutions to computer security problems are actually rather straightforward.

Why Computers Aren't Secure

Most people question why computers are so insecure—after all, people have been hacking for a long time. The vast majority of hacking incidents occur because of one of the following pervasive problems:

Security is an annoyance. Administrators often fail to implement security features in operating systems because doing so causes problems for users. Users also circumvent security—by choosing easy-to-use (easy-to-guess) passwords like "123456," never changing those passwords, disclosing those passwords to co-workers, or sharing user accounts.

Vendors ship software so that it will install in the most feature-filled configuration with its security features disabled so that unskilled users won't run into roadblocks and don't have to understand and configure it correctly before they use it. This means that the vast majority of installations are never properly secured.

The fact that strong security is an annoyance that requires extra learning on the part of everyone involved is the most common reason for security failures.

Features are rushed to market. Vendors concentrate their efforts on adding features that make their software more useful, with little thought to security. A perfect example of this is the addition of scripting language support to Microsoft Outlook and Outlook Express.

virus
Any program that automatically replicates itself.

When the Internet first took off, "e-mail *virus*" scares propagated around the Net via e-mail. Computer security experts ignored them, knowing that a virus required an execution environment like a computer language in order to actually propagate. They laughed at the possibility that anyone would actually tie a computer language to an e-mail system because anyone with any security consciousness at all would never let this happen. Despite the warnings, and even though the scripting language support built in to Microsoft Office had already been exploited to create "macro" viruses embedded in Word and Excel documents, Microsoft ignored the signs and the explicit warnings of its own employees and incorporated a scripting language into its e-mail software. Even worse, it was set up to automatically execute code contained in e-mail messages, configured to do so by default, and included features like "auto-preview" that even opened the messages upon arrival and executed the embedded code. To make matters even more egregious, Microsoft shipped this insecure software for free with every copy of their ubiquitous Windows operating system, thus ensuring that it would be widely deployed.

hacker
One who engages in hacking.

Thus, the plague that is e-mail viruses today arrived—well predicted, forewarned, and completely ignored by a vendor in order to implement a feature that less than 1 percent of legitimate users actually ever use. Microsoft simply didn't concern itself with even a cursory study of the

security implications of adding this feature to its software. It couldn't have done a better job of implementing a new hacking exploit if it had been doing it on purpose.

Vendors who spend time on security are eclipsed by the competition. Customers don't truly value security. If they did, they would use older, well-tested, security-proven software that doesn't have all the bells and whistles of the latest versions. Companies like Microsoft that retrofitted their existing products to work on the Internet decimated their competition. Had they waited to do it securely, they would have been beaten to market by someone who didn't. The end result? The least-secure products always get to market first and become standards.

Computers and software evolve very quickly. Computers and networking technology have been evolving far faster than companies can predict what might go wrong with them. Moore's law states that computer hardware will double in power every two years. His prediction has been eerily accurate for over three decades now.

Protocols that were not developed to be secure were adapted to purposes that they were never intended for and then grew in popularity to a far wider audience than the original creators could have imagined.

Programmers can't accurately predict flaws. Programmers rarely consider that the state of their functions might be externally changed to any possible value while the code is running, so they only check for values that they send to it themselves. Once the code passes its normal debugging checks, it's shipped without having been tested to pass a barrage of random data thrown at it. Even if they did attempt to predict flaws, the 10 programmers who created a project could never come up with the complete set of attacks that the million hackers who attempt to exploit it will.

There is little diversity in the software market. The duopoly of the *Windows* and *Unix* operating systems has narrowed the targets of hackers to minor variations on just two operating systems. In most applications, just one or two products make up the lion's share of the market, so hackers have to crack only one product to gain wide access to many people. Two web servers, Apache and IIS, compose more than 90 percent of the web service market. Two closely related families of operating systems, Windows and Unix, compose more than 90 percent of the operating system market for PCs.

Vendors are not motivated to reveal potential flaws. To avoid marketing fiascoes, vendors try to hide problems with their operating systems and thereby naturally discourage discussion of their flaws. Conversely, hackers publicize flaws they discover immediately to the entire world via the Internet. This dichotomy of discussion means that flaws are far more widely disseminated than the solutions to them are.

Windows
A family of single-user operating systems developed by Microsoft for small computers. The most recent version has incorporated enhancements to allow multiple users to run programs directly on the machine.

Unix
A family of multiuser operating systems that all conform completely to the Portable Operating System Interface for Unix (POSIX) specification and operate in very similar fashion; this includes Unix, BSD, Linux, and derivatives of these major versions.

firewall
A packet router that inspects the data flowing through it to decide which information to pass through based upon a set of programmed policies.

hacking
The act of attempting to gain access to computers without authorization.

protocol
An agreed-upon method of communicating between two computers.

Patches are not widely deployed and can cause problems when they are installed. When security problems are found with a piece of software, the vendor will fix the problem, post a patch on the Internet, and send out an e-mail notice to registered customers. Unfortunately, not everyone gets the notice or installs the patch—in fact, the majority of users never install security patches for software unless they actually get hacked.

Even worse, vendors rush security patches to clients with unexposed bugs that can cause even more serious problems on their client's machines and even in the best cases require additional processing to find the flaws, thus slowing the systems. In some cases, the cure can be worse than the disease.

With these problems epidemic in the security market, you might wonder if the security problem will ever be solved. In fact, there will always be flaws in software. But there are many relatively easy things that can be done to fix these problems. Secure *protocols* can be layered on top of unsecured protocols or replace them outright. Border security with *firewalls* can prevent *hackers* from reaching most systems, thus making their security flaws unimportant. Compilers and computer languages can be modified to eliminate problems that programmers fail to check for. And vendors can find ways to make security more convenient, such as filtering easily guessed passwords using spell-checker technology. And, as hackers continue to exploit systems, customers will demand proactive security and reward vendors who emphasize security rather than those who ship feature-filled, but poorly thought-out, products.

NOTE

Why can't vendors make software secure out of the box? In truth, they can. In the OpenBSD operating system, there has been only one remotely exploitable flaw found in seven years. Its developers have accurately predicted and proactively closed *hacking* exploits before they could be exploited. But OpenBSD is not very popular because it doesn't have a lot of features—it's just a basic operating system, and your own software can still be exploited once you add it.

The History of Computer Security

When you understand the history of computer security, it becomes obvious why computers aren't secure.

worm
Any program that takes active measures to replicate itself onto other machines in a network. A network virus.

Stories of major, nearly catastrophic, hacking exploits happen all the time. 2001 was a particularly bad year for Internet security. The Code Red *worm* spread unchecked through the Internet—and once it was patched, the Nimbda virus did almost exactly the same thing; e-mail viruses spread with regularity, and Microsoft shipped its newest flagship operating system, Windows XP, with a security flaw so egregious that hackers could literally exploit any computer running it with no serious effort at all; the Linux standard FTP and DNS services were exploited, allowing hackers to enter websites and deface their contents at

will. As of 2004, Nimda variants are still prowling the Internet, hitting newly installed machines while cousins like Sasser use the same old propagation code patched to attack new vulnerabilities. It seems like hacking is just getting worse, even as organizations spend more money on the problem. In fact, widespread hacking is getting more common.

In 1988, the year in which reporting began, the Computer Emergency Response Team (CERT) at Carnegie Mellon University, which tracks Internet security incidents, reported six hacking incidents. In 1999, they reported nearly 10,000. In 2000, they reported over 22,000. In 2001, they reported over 52,000 incidents. Numbers like these can sound scary, but when you factor in the growth of the Internet by counting incidents per computers attached to the Internet, security incidents are rising at a rate of 50 percent per year (rather than the 100 percent per year the raw numbers suggest) and have been since 1993, the first year for which reasonably reliable information is available about the overall size of the Internet. A slight decline in the percentage of incidents reported is evident since 2001, with 82,000 incidents in 2002 and 138,000 in 2003, so explosive growth trend appears to be slowing.

The following sections are a quick reprisal of computer security since the dawn of time. (See the graphic on the next page.)

–1945

Computers didn't exist in any real sense before 1945. The original need for security (beyond prevention of outright theft of equipment) sprang from the need for secure military and political communication. *Codes* and *ciphers* were originally studied because they could provide a way to secure messages if the messages were intercepted and could allow for distance communication like smoke, mirror, or pigeon signaling.

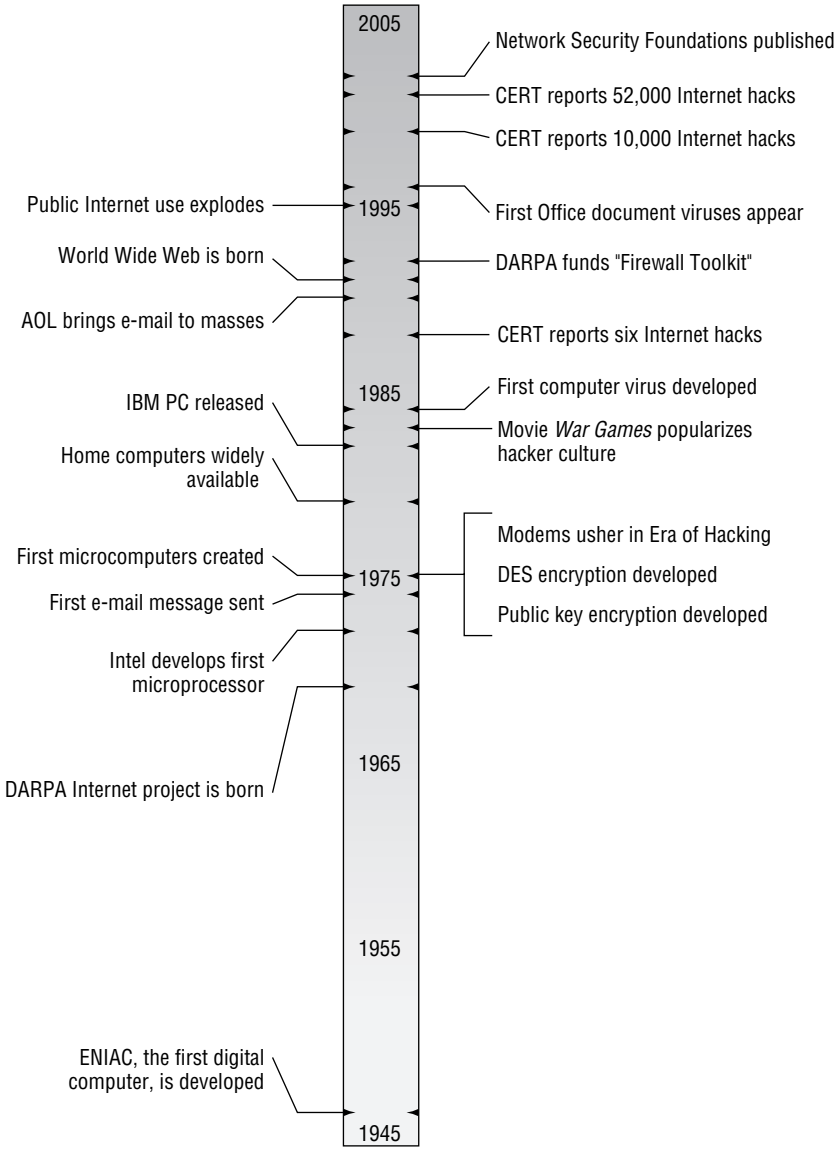
code
An agreed-upon set of symbols that represent concepts. Both parties must be using the same code in order to communicate, and only predetermined concepts can be communicated.

cipher
A mathematical function used to transform a plain message into a form that cannot be read without decoding it. Ciphers can encode any message.

Before the advent of telegraphy, telephony, and radio communications, simply transmitting a message anywhere was extremely difficult. Wars were prosecuted slowly; intrigues were based on hunches, guesses, and paranoia because real information was difficult to come by. Messages transmitted by post or courier were highly likely to be intercepted, and when they were, the consequences were disastrous for the war or political effort.

For that reason, codes, which are far easier to implement than ciphers, formed the backbone of secure communications prior to the advent of automated computing. Codes are simple substitution ciphers—one word is used to transmit another word, concept, or phrase. Both parties encode and decode their messages using a codebook, and generally the codes were chosen so that they made reasonable sense when read in their coded form in an attempt to hide the fact that they were encoded—similar to the modern concept of steganography, or hiding encrypted data as noise inside other content like a digital picture or sound file. (Most militaries

still use codes and codebooks for operational messages over unencrypted radio links as a holdover from earlier times, but as computing power becomes cheap, this practice is quickly fading into obscurity.) Unfortunately, both parties had to have the codebook, and the interception of a codebook meant that all encoded communication could be decoded.



1945–1955

A half-century ago, the first electronic computers were being developed. These gargantuan machines operated on vacuum tubes and had considerably less computing power than today's $50 calculator. They cost many millions of dollars to build and operate, and every compute cycle was precious. Wasting computing time on such luxuries as security was unheard of—but since you had to have both physical access and substantial training to operate these machines, security was not a problem. With so many other problems to solve, computer security wasn't even on the research horizon at this time.

1955–1965

As computers moved into the business world in the sixties, computer security was limited only to making sure that the occasional disgruntled employee couldn't cause harm and that the competition had no access to the computers. Both measures still relied upon physical security for the environment rather than security measures in software. Accounts and passwords, when implemented, were simple and used merely for tracking which users performed which actions in the system rather than for any form of true security. There's not a single verified instance of remote malicious hacking activity occurring during or before this era.

1965–1975

During the late sixties and early seventies, as *mainframes* grew more powerful and the number of users attached to them reached into the thousands, accountability became more important. To limit what typical users could do, the concept of limited user accounts and unlimited administrative accounts came into practice. Typical users could not perform actions that might corrupt data or disrupt other users, while administrators could do anything that was necessary on the system. User accounts protected by passwords were used to discriminate between the various types of users. Most mainframes shipped from the factory with a default password that the administrators were responsible for changing once they received the machine—a practice that is still common with simple network devices.

mainframe
A large and powerful (in context) computer that many users share via terminal displays.

Operating system research was beginning to take root in this period, and mainframe operating systems like Multics were beginning to be adapted to a much smaller breed of business-class machines, like minicomputers and the first single-user systems called workstations. The phone company was involved in a tremendous amount of operating research at the time, and developed a light version of Multics, called Unix. At the same time, Digital Equipment was developing a more portable version of its operating system, called VMS, while IBM worked on its various mainframe operating systems.

operating system
The program that controls the overall operation of a computer.

Hacking in this era consisted of mere rumors of rogue programmers performing illicit hacks—such as writing code that took the fractional remnants of rounded transactions and deposited them in their own bank accounts or writing back doors into their code so that they could always gain access to systems (as the original developers of Unix have insinuated that they did).

1975–1985

The lack of true security came to light in the seventies when companies started providing remote access to terminal users over modems that operated using the public telephone system. Modems allowed small offices to connect directly to central computers in the corporate headquarters. Companies also leased the newer digital phone circuits and began connecting remote offices directly to their systems over "leased lines" that did not require modems and could span the country—at great expense. And, since only direct connections could be made between mainframes and terminals, there was very little flexibility for routing information.

The military had been using computers for years at this point and had been chafing at the lack of flexibility in sending messages between mainframes. In 1969, the Defense Advanced Research Projects Agency (DARPA) initiated a project to explore the promise of packet-based networks, where individual tiny messages could be transmitted between two end systems and routed by intermediate systems connected in a loosely hierarchical method, thus allowing any participants on the network to communicate. These research efforts began to bear useful fruit in the late seventies.

The amount of computing power required to perform message (or packet) routing was impractical at the time, but it was clear that computers would quickly become powerful enough to make the problem trivial in the next few years. Because message routing required intermediate systems to perform work that didn't directly involve them, security was antithetical in the early packet-based research systems; intermediate systems could not waste the time to authenticate every packet that went through them, and requiring security would have kept the system from getting off the ground. But in the military, physical security and accountability more than made up for the lack of systems security, and since no untrusted users were attached to the system, security wasn't an issue.

Data Encryption Standard (DES)
A secret key encryption algorithm developed by IBM, under contract to the U.S. government, for public use.

But the government realized that security would become an issue and began funding major initiatives to improve computer security. IBM developed the *Data Encryption Standard (DES)* for the government in 1975. And at nearly the same time, Whitfield Diffie and Martin Hellman developed the concept of the *public key encryption (PKE),* which solved the longstanding problem of secure key exchange. In 1977, Rivest, Shamir, and Adelman implemented PKE in the proprietary RSA encryption algorithm. These pioneering efforts in network encryption weren't widely deployed at the time, but they are the foundation of computer security today.

The development of the microprocessor by Intel in 1972 was beginning to bear fruit: four or five models were available to the public by 1975. Hobbyists

could build their own computers from parts available through catalogs, and by 1978 complete computer systems could be purchased off the shelf by end users in any town in the U.S.

They could be purchased with modems that were capable of communicating directly with corporate computers as well, and the art and practice of hacking was born.

Hacking in those days consisted of "war-dialing" a range of phone numbers automatically by leaving hobby computers running overnight. Whenever a computer answered, the computer doing the war-dialing would typically print out the phone number. In any case, it would hang up immediately, causing numerous nuisance calls to people in the middle of the night. The hacker would then go through the list of found computers manually, looking for signs of computers that might be easy to break into, like mainframe computers whose default administrative *passwords* had never been changed.

public key encryption (PKE)
A method of encryption that solves the problem of exchanging secret keys by using different but related ciphers for encoding and decoding.

password
A secret known to both a system and a user that can be used to prove a user's identity.

After a few high-profile, apparently effortless cases of hackers breaking into computer systems occurred, the concept of *call-back security,* also known as dial-back security, was introduced. With call-back security, the answering computer (the system) accepts only a phone number from the calling computer (the client) and hangs up. The system then checks this phone number against an allowed list, and if it appears, the system calls back the client whose computer is set to listen for a call back. The fact that phone numbers can't easily be forged and that phone lines are somewhat difficult to tap made for all the security that was necessary in those days.

call-back security
Security that is implemented by having the main system call the remote user back, thus ensuring that the user attempting to gain access is an authorized one (so long as the phone system remains secure).

Hackers did have the ability to hack the telephone company's computers to reroute phone calls and manually direct where calls went, but hackers with these skills were extremely rare, and lacking any public discussion forum, every hacker pretty much had to learn these techniques on their own. By the mid-eighties, call-back security had solved the problem of computer security to the point that it was worth solving, and increased security by the public telephone companies made exploiting these systems very difficult.

1985–1995

bulletin-board system (BBS)
A single central computer to which many computers have intermittent access to shared information.

In the mid-eighties, the popularity of PC computers exploded; PCs went from a novelty owned by geeks to an essential tool of nearly every desktop in the country in the span of 10 years. With the explosion in popularity grew the need to connect PC computers together directly, and so local area networks, pioneered in the previous decade, came out of the research closet and onto the desktop as well. These networks used business-grade versions of the military's packet-based networks that were optimized for small networks. By 1995, networked PCs were crucial to the business world.

At the same time, home computer enthusiasts with modems were creating online communities called *bulletin-board systems (BBS)*. By using a single expensive PC with a lot of modems or an obsolete mainframe as a central server, home

users could dial in to chat with friends, send text messages, and participate in online discussion groups and games. Without exception these services were text-based to make maximum use of the slow modem links and low processing power of the computers of the day.

Some of these BBSs became very large. CompuServe became the largest BBS at this time, linking millions of computer users by modem and allowing them to trade electronic mail and to "chat" or use text messages with one another in real time. Another company, America Online, took the BBS concept and put a graphical interface on it, making getting "on line" easy enough for computer novices.

BBSs allowed hackers to begin trading in information and to form distributed hacking cabals—usually targeting other BBSs because most business computers had become locked down with the advent of dial-up security. Hacking in this period worked largely the same way that it had in the seventies except that the targets were new phone companies, BBSs, and the occasional improperly secured corporate mainframe.

That is, unless you happened to be a student at a university. During these years, universities took over development of the military's original packet-routing protocols and developed them to solve real-world problems. Just like the military prototype, these systems relied on the fact that intermediate systems would route data without authentication in order to function. Security was a layer pasted on top, in the actual application that used the packet network, rather than at the network layer. This allowed clever students to watch data flowing through intermediate systems to gather passwords and then use those passwords to gain access to other systems. Because military installations and academic research companies were also connected to this "Internet," early hackers had the chance to cause real mischief—but rarely actually did.

During this period, e-mail grew out of simple messaging systems that allowed only interoffice communication into a messaging system that could span companies and allow anyone attached to the Internet to trade real, human information. Other research projects like FTP and Gopher allowed people to trade computer files and documents over the Internet. In 1990, Gopher was merged with a research concept called HyperText (previously seen by the public in Apple's HyperCard product) to produce "browsable documentation" that contained embedded links to other documents that could be automatically downloaded when the link was selected. This technology, called the World Wide Web, allowed scientists to publish their scientific papers immediately and was an immediate boon to the scientific and Internet computing communities.

The fact that hacking could occur on the nascent Internet didn't pass unnoticed, however. Every major entity attached to the Internet, including the military, universities, and mainframe computer companies like IBM and Digital, developed special intermediate systems that performed extra analysis of data flowing through them to determine if the data was legitimate and should be routed. These routers were called firewalls.

1995–2005

The Internet exploded on the public scene between late '94 and early '96 (we'll just call it '95). Borne largely by the twin utilities of universal e-mail and the World Wide Web, the Internet became so compelling that the owners of most BBSs began to connect their systems to the Internet and the government turned over management of it to a consortium of Internet service providers (ISPs). Universities frequently allowed wide access to their Internet connections early on, and soon, phone companies began installing pure "modem banks" to answer phone connections and put them directly on the Internet. The Universities, BBS operator, and phone companies all became Internet service providers, and within an amazingly short period of time, millions of people were connected directly to one another over the Internet. BBSs who didn't convert to ISPs, with the solitary exception of AOL (which provided a bridge to the Internet but maintained its proprietary BBS client software), became extinct almost overnight.

The Internet boom happened so fast that software vendors were caught completely off guard. Bill Gates, the chairman of Microsoft, said in 1994 that the Internet would blow over. His words merely echoed the typical response of most PC software developers. Some new companies, like Netscape, consisted of students who had been using the Internet at school and knew its potential, but these companies were few and far between.

By the next year, it was obvious that the Internet wasn't going to just blow over. In a telling incident, Mr. Gates called a meeting at his retreat and forced his entire staff to abandon their current developments and refocus their efforts on making every one of Microsoft's products "Internet Enabled." Other software companies couldn't react as quickly, and the Internet caused many of them to stumble, ship late, and become irrelevant. Only those who rushed to make their software and operating systems compatible with Internet protocols remained in the game. The very largest names in computer software at the time, like Borland, WordPerfect, Novell, IBM, and Lotus, were all simultaneously hobbled by the fact that Microsoft was able to make its products take advantage of this new technology in short order, while they chose to finish their current developments and wait for the next development cycle to make their products Internet-ready. By the time their next product revisions came out, nobody cared and Microsoft had completely eclipsed them all.

The rush to market, while a marketing coup for Microsoft, made security an afterthought. The folks at Microsoft actually believed their own hype about their flagship operating system, Windows NT, and felt that its office-grade security would make it the most secure operating system on the Internet. For their home use products like Windows 95, 98, and Me, security wasn't even attempted—you could gain access to the computer by clicking "cancel" at the log-in dialog, if one was even configured to appear. After all, if Microsoft had held up the development of these products to try to make them secure, end users would have just adopted somebody else's insecure products that were ready to go.

The Internet, with its totally nonsecure protocols, was the fertilizer that the hacking world needed after the sparse desert of the late eighties. Once phone companies had locked down their systems, hacking had frankly become rather boring and routine. Anybody you could hack wasn't going to be interesting anyway, so there was little point in trying. But suddenly, everyone was attached to the same insecure network, ripe for the plucking.

Microsoft's dominance of the PC software market meant that hackers could concentrate their efforts on understanding just two operating systems: Unix, the native OS of the Internet, and Windows, the operating system of the masses. By creating exploits to hack these two operating systems remotely over the Internet, hackers gained almost unlimited access to information on the Internet. Vendors scrambled to patch security problems as soon as they were discovered, but the lag between discovery and response left weeks during which hackers could broadcast their discoveries and cause widespread damage.

Businesses clamped down by installing firewalls, evolved from early military and commercial security research efforts, onto their leased lines at the point where they attached to their ISPs. Firewalls went a long way toward protecting interior systems from exploitation, but they still allowed users to circumvent security accidentally and did little to stop the exploitation of services that had to be allowed—like e-mail and web services. These two services now constitute the bulk of hacking targets because they can't be blocked while still operating correctly.

encryption
The process of encoding a message using a cipher.

Toward the close of this era, *encryption* gained wide use as the savior of the Internet. By implementing security protocols that could hide data and prove someone's identity while preserving the ease-of-use and ubiquity that made the Internet popular, encryption, along with firewalling, is basically saving the Internet from abandonment due to security concerns.

Hackers will continue to exploit insecure protocols, but as vendors learn to ship secure software or shore it up with integrated firewall code, and as implementers learn to secure their own systems, hacking is doomed to drift steadily toward the situation in the late eighties, when it was no longer that interesting because those remaining insecure users were trivial.

2005–

Hacking will drop off dramatically once Microsoft integrates strong firewalling software into all of its operating systems, which will occur late in 2004 when it realizes that the adoption of its new e-commerce .NET services depends upon security rather than features. The open-source community and their flagship Linux product had already integrated true firewalling years earlier, and Linux is seen as more secure than Windows—a situation that Microsoft will not tolerate for long. Apple will simply adapt the open-source firewalling services into Mac OS X, which is based upon BSD Unix, to prevent its exploitation, and every other commercial version of Unix will be completely eclipsed and made

obsolete by the free, faster moving, and more secure Linux or BSD Unix operating systems by this time.

E-mail forgery and spamming will become more popular, until users begin to use the X.509 certificate-based encryption and digital signature capabilities already supported but rarely used. Someone (probably Microsoft, Yahoo, or AOL) will set up a free certificate authority for private users and make mail clients and web browsers automatically download certificates from it as part of an online digital identity that will be used to enable secure e-commerce services.

Once Microsoft and the open-source community tighten down the hatches on their operating systems and services, hacking exploits will become fewer and farther between. The government will catch up with hacking activity after it tapers off and begin making examples of people again. Hacking as a hobby will taper down to a trickle.

Until a researcher somewhere and somewhen discovers a fundamental mathematical flaw in the encryption software upon which all of these security measures are based…

Security Concepts

Computer security is based on the same concepts that physical security is: trust, knowledge of a secret to prove authenticity, possession of a key to open locks, and legal accountability. The metaphors are so apt that most computer security mechanisms even have the same names as their physical counterparts.

Trust

All computer security springs from the concept of inherent or original trust. Just as a child inherently trusts its parents, a secure computer system inherently trusts those who set it up. While this may seem rather obvious, it is an important concept because it is the origination of all subsequent security measures.

There's more inherent trust in computer security than simply the original establishment of a system. For example, you trust that there are no "back doors" in the software you use that could be exploited by a knowledgeable person to gain access. You trust that the login screen that you are looking at is actually the system's true login screen and not a mimic designed to collect your password and then pass it to a remote system. Finally, you trust that the designers of the system have not made any serious mistakes that could obviate your security measures.

Authentication

Authentication is the process of determining the identity of a user. Forcing the user to prove that they know a secret that should be known only to them proves that they are who they say they are.

authentication
The process of determining the identification of a user.

user account
A record containing information that identifies a user, including a secret password.

smart card
An electronic device containing a simple calculator preprogrammed with a code that cannot be retrieved. When given a challenge, it can calculate a response that proves it knows the code without revealing what the code is.

User accounts are associated with some form of secret, such as a password, PIN, biometric hash, or a device like a *smart card* that contains a larger, more secure password than a human could remember. To the system, there is no concept of a human; there is only a secret, information tied to that secret, and information to which that secret has access.

Authentication is only useful in so far as it is accurate. Passwords are probably the least reliable form of authentication in common use today, but they're also the most easily implemented—they require no special hardware and no sophisticated algorithms for basic use. However, they are easily guessed, and even when they're carefully chosen it's still possible to simply guess the entire range of possible passwords on many systems in short order.

A less common but more secure method of authentication is to physically possess a unique key. This is analogous to most physical locks. In computer security systems, "keys" are actually large numbers generated by special algorithms that incorporate information about the user and are stored on removable media like smart cards. The problem with keys is that, like physical keys, they can be lost or stolen. However, when combined with a password, they are very secure and difficult to thwart.

Another form of authentication provides inherent identification by using a physical property of the user. This is called biometric authentication, and it relies upon unique and unchangeable physical properties of a human, such as handwriting characteristics, fingerprints, facial characteristics, and so forth. Biometric authentication has the potential to be the most reliable form of authentication because it's easy to use, nearly impossible to fake when correctly implemented, and can't be circumvented for the sake of convenience. Some forms of biometric authentication are easier to "forge" than others, and naïve implementations can sometimes be easily faked. But when well implemented, biometric authentication is the most secure form of authentication and the only form that can be truly said to uniquely and unmistakably identify a user.

Chain of Authority

trust provider
A trusted third party that certifies the identity of all parties in a secure transaction. Trust providers do this by verifying the identity of each party and generating digital certificates that can be used to determine that identity. Trust providers perform a function analogous to a notary public.

During the installation of a security system, the original administrator will create the root account. From the root account (called the "administrator" account in Windows and the "Supervisor" account in NetWare), all other accounts, keys, and certificates spring. Every account on a system, even massive systems containing millions of accounts, spring from this chain of authority. The concept of chains of authority isn't often discussed because it is inherent in a secure system.

Certificate systems are also based on a chain of authority. Consider the case of separate businesses that do a lot of work together. It would be convenient if users from Business Alpha could automatically log on to computers at Business Beta. But because these two systems have two different chains of authority, there's no way for Business Alpha to trust that users who say they are from Business Beta

actually are. This problem is solved by having both businesses trust a third-party *trust provider,* or a company that specializes in verifying identity and creating secure certificates that can be used to prove identity to foreign systems. As long as both businesses trust the same trust provider, they are rooted in the same chain of authority and can trust certificates that are generated by that trust provider. Trust providers are the digital equivalent of a notary public. Examples of trust providers are VeriSign and Thawte.

Accountability

Accountability is where the secret meets the user. Users don't try to circumvent security because their identity would be known and they would be held legally accountable for their actions. It is accountability, rather than access controls, that prevents illegal behavior.

In pure accountability-based systems, no access control mechanisms are present. Users simply know that their every action is being logged, and since their identity is known and their activities are tracked, they won't do things that could jeopardize their position (unless something happens to make them no longer care).

The problem with accountability-based systems is twofold—they only work if identity can't be faked, and there are rare occasions where users lose their inhibitions. Without access control, these users can destroy the entire system. For these reasons, accountability-based security is normally used to augment access control systems rather than to replace them.

Access Control

Access control is the security methodology that allows access to information based on identity. Users who have been given permission or keys to information can access it—otherwise, access is denied.

Permissions-Based Access Control

Once the system knows the identity of an individual because they've been authenticated, the system can selectively allow or deny access to resources like stored files based on that identity. This is called permissions-based security because users are either granted or denied permission to access a *file* or other resource.

file
A sequence of related information referenced by a filename in a directory.

The question of who has access to which files is typically either defined by administrators when the system is implemented or created according to some set of default rules programmed into the system; for instance, the original creator (owner) of a file is the only user who can change it.

Access controls are typically implemented either as directory permissions that apply to all files within the directory or by an access control list, which is a component of a file that explicitly lists which users can access it. Typically, when a

file is created, an ACL is automatically copied from the parent directory's ACL, so it is said to "inherit" permissions from the containing directory.

Unfortunately, none of these security controls works if the operating system can be circumvented. By shutting off the system and mounting its storage in another computer, a foreign system can read off all the files without interference because it's not asking for permission from the operating system. Essentially, permissions can be circumvented the same way kids can disobey their parents—by simply not asking for permission in the first place.

Encryption-Based Access Control (Privacy)

private key
The key used to decode public key messages that must be kept private.

A totally different way to control access is to simply encrypt data using public key encryption. Access to the encrypted data is given to those who want it, but it's worthless to them unless they have the *private key* required to decode it. Using PKE to secure data works very well, but it requires considerably more processing power to encode and decode data.

TIP

Encryption is such an important topic in computer security that it requires its own chapter to be covered properly. If you don't understand the terms used in this section, just reread it after you read Chapter 3.

Encryption-based access control is also dangerous because data can be irrevocably lost if the private key required to decrypt it is lost. For this reason, most practical systems store a copy of a resource's private key in a key repository that can be accessed by an administrator, and the copy itself is encrypted using another key. The problem of potential loss of information doesn't go away, but the system includes more participants and therefore permanent loss is less likely to happen.

Practical systems also don't encrypt files with a unique public key for each file or user—in fact, they encrypt files using a secret key registered to an entire group and then encrypt the list of secret keys for the group using a private key. The private key is given to each member of the group (possession of the private key makes one a member of the group). Thus, members of the group have the key to decrypt the store that contains the secret key required to decrypt the file. This way, when an account is deleted, no keys are irrevocably lost because other members still have the key.

In pure encryption-based access control systems, the keys for a group are stored in a file that is encrypted using a user's smart card. By possessing the smart card, a user can decrypt the store that contains the keys for the groups they are members of, and those keys in turn can be used to decrypt the store that contains the keys that are used to decrypt individual files. This is how a chain of authority is created using encryption, and systems that work this way are called Public Key Infrastructure (PKI) systems.

No common systems work this way yet, but support for PKI is being retrofitted into both Windows and Unix. Shortly, most systems will work this way.

Encryption-based access control solves the problem of requiring the operating system to arbitrate access to secure data. Even if the operating system has been circumvented, stored data is still encrypted. Encrypted data can be transmitted over public media like the Internet without concern for its privacy.

Terms to Know

authentication
bulletin-board systems (BBS)
call-back security
ciphers
codes
Data Encryption Standard (DES)
encryption
file
firewalls
hackers
hacking
mainframes
operating system
passwords
private key
protocols
public key encryption (PKE)
smart card
trust provider
Unix
user accounts
virus
Windows
worm

Review Questions

1. What is security?
2. What is the most common reason security measures fail?
3. Why would vendors release a product even when they suspected that there could be security problems with the software?
4. How many operating systems make up 90 percent of the operating system market?
5. Factoring in the growth of the Internet, at what rate is the number of computer security incidents increasing?
6. Why weren't computers designed with security in mind from the beginning?
7. During what era did "hacking" begin to occur en masse?
8. In what year was public key encryption developed?
9. Prior to the Internet, how did most hackers share information?
10. Why is it likely that applications (other than those designed to implement security) that concentrate on security will fail in the marketplace?
11. What is the process of determining the identity of a user called?
12. When a new computer is first set up, how does the system know that the person setting up the computer is authorized to do so?
13. What is the most secure form of authentication?
14. How can a hacker circumvent permissions-based access control?
15. How can a hacker circumvent correctly implemented encryption-based access control?