CHAPTER 15

Attacks and Malware

Upon completion of this chapter, you will be able to:

- Describe various types of computer and network attacks, including Denial-of-Service, spoofing, hijacking, and password guessing
- Describe the different types of malicious software that exist, including viruses, worms, Trojan horses, and logic bombs
- Explain how social engineering can be used as a means to gain access to computers and networks
- Explain the importance of auditing and what should be audited

While hackers and viruses receive the most attention in the news (due to the volume of these forms of attack), they are not the only methods for attacking computer systems and networks. This chapter addresses the many different ways computers and networks come under attack on a daily basis. Each type of attack threatens at least one of the three security requirements mentioned in Chapter 1—confidentiality, integrity, and availability (the CIA of security). Attacks are thus attempts by unauthorized individuals to access or modify information, to deceive the system so that an unauthorized individual can take over an authorized session, or to disrupt service to authorized users.

Attacking Computer Systems and Networks

From a high-level standpoint, attacks on computer systems and networks can be grouped into two broad categories: attacks on specific software (such as an application or the operating system itself) and attacks on a specific protocol or service. Attacks on a specific application or operating system are generally possible because of either an oversight in the code (and possibly in the testing of that code) or because of a flaw or bug in the code (again indicating a lack of thorough testing). Attacks on specific protocols or services are attempts to either take advantage of a specific feature of the protocol or service or to use the protocol or service in a manner for which it was not intended. In addition, the target of an attacker can be of two types: targets of opportunity where the attacker is attempting to find any machine that is susceptible to a specific vulnerability, and defined targets where the attacker wants to gain access to a very specific target and must attempt to find a vulnerability that exists in that target. The remainder of this section discusses the various forms of attacks that security professionals need to be aware of.

Denial-of-Service Attacks

Denial-of-Service (DOS) attacks can exploit a known vulnerability in a specific application or operating system, or they may attack features (or weaknesses) in specific protocols or services. In this form of attack, the attacker is attempting to deny authorized users access either to specific information or to the computer system or network itself.

The purpose of such an attack can be to simply prevent access to the target system, or the attack can be used in conjunction with other actions in order to gain unauthorized access to a computer or network. For example, a SYN flooding attack may be used to temporarily prevent service to a system in order to take advantage of a trusted relationship that exists between that system and another.

SYN flooding is an example of a DOS attack that takes advantage of the way TCP/IP networks were designed to function, and it can be used to illustrate the basic principles of any DOS attack. SYN flooding utilizes the TCP three-way handshake that is used to establish a connection between two systems. Under normal circumstances, the first system sends a SYN packet to the system it wishes to communicate with. The second system will respond with a SYN/ACK if it is able to accept the request. When the initial system receives the SYN/ACK from the second system, it responds with an ACK packet, and communication can then proceed. This process is shown in Figure 15-1.

NOTE A SYN/ACK is really a combination of the SYN packet sent to the first system, combined with an ACK packet acknowledging the first system's SYN packet.

In a SYN flooding attack, the attacker sends fake communication requests to the targeted system. Each of these requests will be answered by the target system, which then waits for the third part of the handshake. Since the requests are fake (a nonexistent IP address is used in the requests, so the target system is responding to a system that doesn't exist), the target will wait for responses that will never come, as shown in Figure 15-2. The target system will drop these connections after a specific time-out period, but if the attacker sends requests faster than the time-out period eliminates them, the system will quickly be filled with requests. The number of connections a system can support is finite, so when more requests come in than can be processed, the system will soon be reserving all its connections for fake requests. At this point, any further requests are simply dropped (ignored), and legitimate users who want to connect to the target system will not be able to. Use of the system has thus been denied to them.

Another simple DOS attack is the famous ping-of-death (POD), and it illustrates the other type of attack—one targeted at a specific application or operating system, as opposed to SYN flooding, which targets a protocol. In the POD attack, the attacker sends

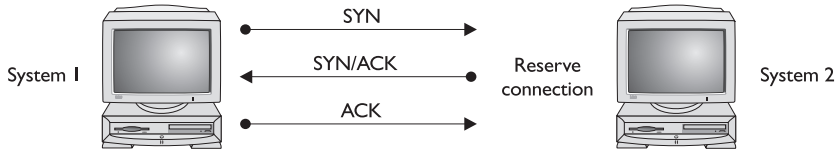

Figure 15-1 The TCP three-way handshake

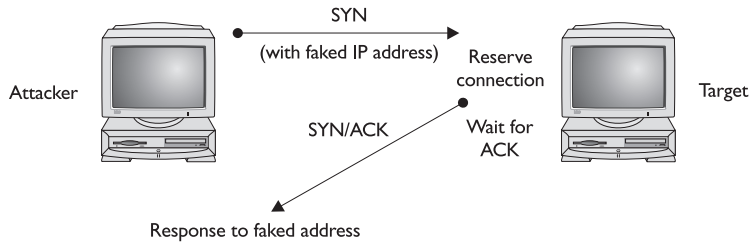

Figure 15-2 A SYN flooding DOS attack

an Internet Control Message Protocol (ICMP) "ping" packet equal to, or exceeding 64KB (which is to say, greater than 64 * 1024 = 65,536 bytes). This type of packet should not occur naturally (there is no reason for a ping packet to be larger than 64KB). Certain systems were not able to handle this size of packet, and the system would hang or crash.

DOS attacks are conducted using a single attacking system. A Denial-of-Service attack employing multiple attacking systems is known as a distributed Denial-of-Service (DDOS) attack. The goal of a DDOS attack is the same: to deny the use of or access to a specific service or system. DDOS attacks were made famous in 2000 with the highly publicized attacks on eBay, CNN, Amazon, and Yahoo.

In a DDOS attack, the method used to deny service is simply to overwhelm the target with traffic from many different systems. A network of attack agents (sometimes called zombies) is created by the attacker, and upon receiving the attack command from the attacker, the attack agents commence sending a specific type of traffic against the target. If the attack network is large enough, even ordinary web traffic can quickly overwhelm the largest of sites, such as the ones targeted in 2000.

Creating a DDOS network is not a simple task. The attack agents are not willing agents—they are systems that have been compromised and on which the DDOS attack software has been installed. In order to compromise these agents, the attacker has to have gained unauthorized access to the system or tricked authorized users to run a program that installed the attack software. The creation of the attack network may in fact be a multistep process in which the attacker first compromises a few systems that are then used as handlers or masters, and which in turn compromise other systems. Once the network has been created, the agents wait for an attack message that will include data on the specific target before launching the attack. One important aspect of a DDOS attack that should be mentioned is that with just a few messages to the agents, the attacker can have a flood of messages sent against the targeted system. Figure 15-3 illustrates a DDOS network with agents and handlers.

How can you stop or mitigate the effects of a DOS or DDOS attack? One important precaution is to ensure that you have applied the latest patches and upgrades to your systems and the applications running on them. Once a vulnerability is discovered, it does not take long before multiple exploits are written to take advantage of it. Generally, you will have a small window of opportunity in which to patch your system between the time a vulnerability is discovered and the time exploits become widely available.

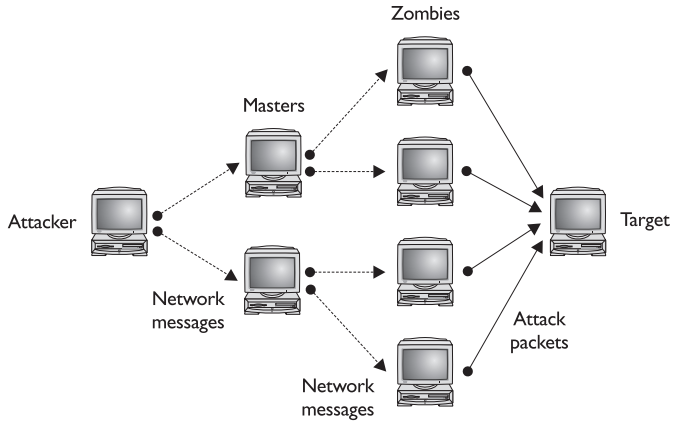

Figure 15-3 Distributed denial of service attacks

Another approach involves changing the timeout option for TCP connections so that attacks such as the SYN flooding attack, described previously, are harder to perform because unused connections are dropped more quickly.

For DDOS attacks, much has been written about distributing your own workload across several systems so that any attack against your system would have to target several hosts in order to be completely successful. While this is true, if large enough DDOS networks are created (with tens of thousands of zombies, for example) any network, no matter how much the load is distributed, can be successfully attacked. This approach also involves an additional cost to your organization in order to establish this distributed environment. Addressing the problem in this manner is actually an attempt to mitigate the effect of the attack, as opposed to preventing or stopping an attack.

In order to prevent a DDOS attack, you have to either be able to intercept or block the attack messages or keep the DDOS network from being established in the first place. Tools have been developed that will scan your systems, searching for sleeping zombies waiting for an attack signal. The problem with this type of prevention approach, however, is that it is not something you can do to prevent an attack on your network—it is something you can do to keep your network from being used to attack other networks or systems. You have to rely on the rest of the community to test their own systems in order to prevent attacks on yours.

A final option you should consider that will address several forms of DOS and DDOS attacks is to block ICMP packets at your border, since many attacks rely on ICMP. Careful consideration should be given to this approach, because it will also prevent the use of some possibly useful troubleshooting tools.

Backdoors and Trapdoors

Backdoors were originally (and sometimes still are) nothing more than methods used by software developers to ensure that they could gain access to an application even if something were to happen in the future to prevent normal access methods. An example

would be a hard-coded password that could be used to gain access to the program in the event that administrators forgot their own system password. The obvious problem with this sort of backdoor (also sometimes referred to as a *trapdoor*) is that, since it is hard-coded, it cannot be removed. Should an attacker learn of the backdoor, all systems running that software would be vulnerable to attack.

The term *backdoor* is also, and more commonly, used to refer to programs that attackers install after gaining unauthorized access to a system to ensure that they can continue to have unrestricted access to the system, even if their initial access method is discovered and blocked. Backdoors can also be installed by authorized individuals inadvertently, should they run software that contains a Trojan horse (more on these later in this chapter). Common backdoors include NetBus and Back Orifice. Both of these, if running on your system, will allow an attacker remote access to your system—access that allows them to perform any function on your system. A variation on the backdoor is the *rootkit*, and they are established not to gain root access but rather to ensure continued root access. Rootkits are generally installed at a lower level, closer to the actual kernel level of the operating system.

Sniffing

The group of protocols that make up the TCP/IP suite was designed to work in a friendly environment where everybody who connected to the network used the protocols as they were designed. The abuse of this friendly assumption is illustrated by network-traffic sniffing programs, sometimes referred to as *sniffers*.

A network **sniffer** is a software or hardware device that is used to observe traffic as it passes through a network on shared broadcast media. The device can be used to view all traffic, or it can target a specific protocol, service, or even string of characters (for example, looking for logins). Normally, the network device that connects a computer to a network is designed to ignore all traffic that is not destined for that computer. Network sniffers ignore this friendly agreement and observe all traffic on the network, whether destined for that computer or others, as shown in Figure 15-4. A network card that is listening to all network traffic and not just its own is said to be in "promiscuous mode." Some network sniffers are designed not just to observe all traffic but to modify traffic as well.

Network sniffers can be used by network administrators for monitoring network performance. They can be used to perform traffic analysis, for example, in order to determine what type of traffic is most commonly carried on the network and to determine which segments are most active. They can also be used for network bandwidth analysis and to troubleshoot certain problems (such as duplicate MAC addresses).

Network sniffers can also be used by attackers to gather information that can be used in penetration attempts. Information such as an authorized user's username and password can be viewed and recorded for later use. The contents of e-mail messages can also be viewed as the messages travel across the network. It should be obvious that administrators and security professionals will not want unauthorized network sniffers on their networks because of the security and privacy concerns they introduce. Fortunately, in order for network sniffers to be most effective, they need to be on the internal network, which generally means that the chances for outsiders to use them against you is extremely limited.

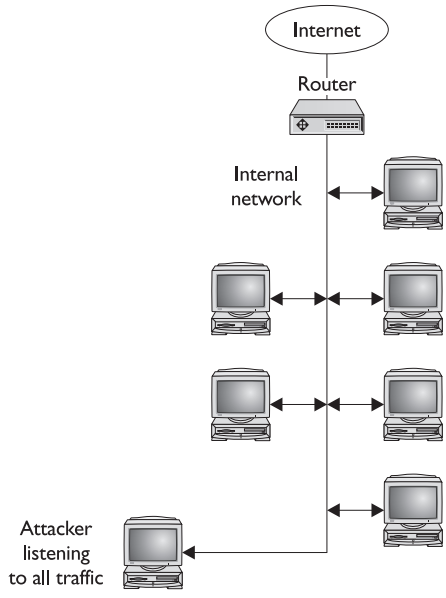

Figure 15-4 Network sniffers listen to all network traffic

Spoofing

Spoofing is nothing more than making data look like it has come from a different source. This is possible in TCP/IP because of the friendly assumptions behind the protocols. When the protocols were developed, it was assumed that individuals who had access to the network layer would be privileged users who could be trusted.

When a packet is sent from one system to another, it includes not only the destination IP address and port but the source IP address as well. You are supposed to fill in the source with your own address, but there is nothing that stops you from filling in another system's address. This is one of the several forms of spoofing.

Spoofing E-mail

E-mail spoofing is where you send a message with a From address different than your own. This can be easily accomplished, and there are several different ways to do it and programs that can assist you in doing so. A very simple method often used to demonstrate how simple it is to spoof an e-mail address is to telnet to port 25 (the port associated with e-mail) on a system. From there, you can fill in any address for the From and To sections of the message, whether or not the addresses are yours and whether they actually exist or not.

There are some simple ways to determine that an e-mail message was probably not sent by the source it claims to have been sent from, but most users do not question their e-mail and will accept where it appears to have come from. A variation on e-mail spoofing, though it is not technically spoofing, is for the attacker to acquire a URL close to the one they want to spoof so that e-mail sent from their system appears to have come from the official site unless you read the address carefully. For example, if attackers wanted to

spoof XYZ Corporation, which owned XYZ.com, the attackers might gain access to the URL XYZ.Corp.com. An individual receiving a message from the spoofed corporation site would not normally suspect it to be a spoof but would take it to be official. This same method can be, and has been, used to spoof web sites. The most famous example of this is probably www.whitehouse.com. The www.whitehouse.gov site is the official site for the White House. The www.whitehouse.com URL takes you to a pornographic site. In this case, nobody is likely to take the pornographic site to be the official government site, and it was not intended to be taken that way. If, however, the attackers made their spoofed site appear similar to the official one, they could easily convince many viewers that they were at the official site.

IP Address Spoofing

The way the IP protocol is designed to work is to have the originators of any IP packet include their own IP address in the "From" portion of the packet. While this is the intent, there is nothing that prevents a system from inserting a different address in the "From" portion of the packet. This is known as IP Address Spoofing. An IP address may be spoofed for several reasons. In a specific DOS attack known as a *smurf* attack, the attacker sends a spoofed packet to the broadcast address for a network, which distributes the packet to all systems on that network. In the smurf attack, the packet sent by the attacker to the broadcast address is an echo request with the From address forged so that it appears that another system (the target system) has made the echo request. The normal response of a system to an echo request is an echo reply, and it is used in the ping utility to let a user know if a remote system is reachable and is responding. In the smurf attack, the request is sent to all systems on the network, so all will respond with an echo reply to the target system, as shown in Figure 15-5. The attacker has sent one packet and has been able to generate as many as 254 responses aimed at the target. Should the attacker send several of these spoofed requests, or send them to several different networks, the target can quickly become overwhelmed with the volume of echo replies it receives.

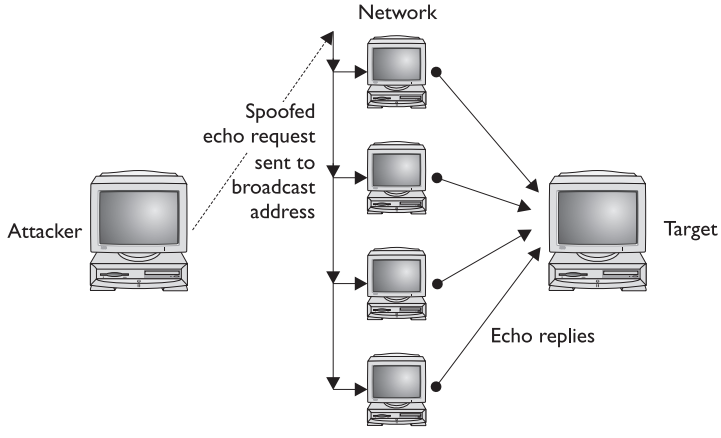

Figure 15-5
Spoofing used in a smurf DOS attack

Spoofing and Trusted Relationships

Spoofing can also take advantage of a trusted relationship between two systems. If two systems are configured to accept the authentication accomplished by each other, an individual logged on to one system might not be forced to go through an authentication process again to access the other system. An attacker can take advantage of this arrangement by sending a packet to one system that appears to have come from a trusted system. Since the trusted relationship is in place, the targeted system may perform the requested task without authentication.

Since a reply will often be sent once a packet is received, the system that is being impersonated could interfere with the attack, since it would receive an acknowledgment for a request it never made. The attacker will often initially launch a DOS attack (such as a SYN flooding attack) to temporarily take out the spoofed system for the period of time that the attacker is exploiting the trusted relationship. Once the attack is completed, the DOS attack on the spoofed system would be terminated and possibly, apart from having a temporarily nonresponsive system, the administrators for the systems may never notice that the attack occurred. Figure 15-6 illustrates a spoofing attack that includes a SYN flooding attack.

Because of this type of attack, administrators are encouraged to strictly limit any trusted relationships between hosts. Firewalls should also be configured to discard any packets from outside of the firewall that have From addresses indicating they originated from inside the network (a situation that should not occur normally and that indicates spoofing is being attempted).

Spoofing and Sequence Numbers

How complicated the spoofing is depends heavily on several factors, including whether the traffic is encrypted and where the attacker is located in relationship to the target. Spoofing attacks from inside a network, for example, are much easier to perform than attacks from outside of the network because the inside attacker can observe the traffic to and from the target and can do a better job of formulating the necessary packets.

Formulating the packets is more complicated for external attackers because there is a sequence number associated with TCP packets. A sequence number is a 32-bit number established by the host that is incremented for each packet sent. Packets are not guaranteed to be received in order, and the sequence number can be used to help reorder packets as they are received and to refer to packets that may have been lost in transmission.

In the TCP three-way handshake discussed previously, two sets of sequence numbers are created, as shown in Figure 15-7. The first system chooses a sequence number to

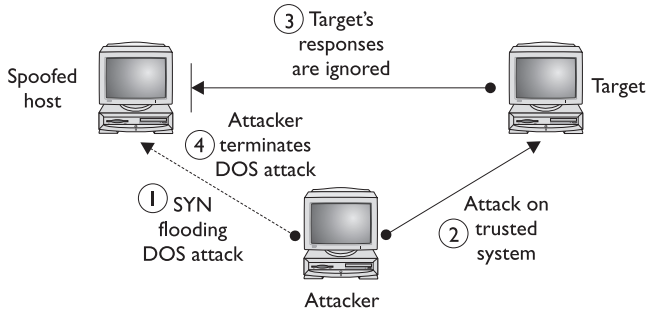

Figure 15-6 Spoofing to take advantage of a trusted relationship

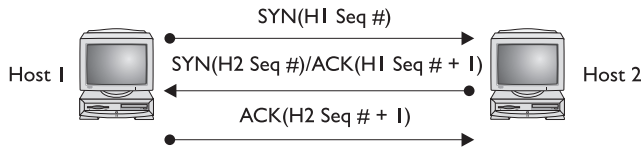

Figure 15-7 Three-way handshake with sequence numbers

send with the original SYN packet that it sends. The system receiving this SYN packet acknowledges with a SYN/ACK. It sends back the first sequence number plus one (that is, it increments the sequence number sent to it by one). It then also creates its own sequence number and sends that along with it. The original system receives the SYN/ACK with the new sequence number. It increments the sequence number by one and uses it in an ACK package it responds with.

The difference in the difficulty of attempting a spoofing attack from inside a network and from outside involves determining the sequence number. If the attacker is inside of the network and can observe the traffic the target host responds with, the attacker can easily see the sequence number the system creates and can respond with the correct sequence number. If the attacker is external to the network, the sequence number the target system generates will not be observed, making it hard for the attacker to provide the final ACK with the correct sequence number. What the attacker has to do is guess what the sequence number might be.

Predicting sequence numbers is possible, because sequence numbers are somewhat predictable. Sequence numbers for each session are not started from the same number, so that different packets from different concurrent connections will not have the same sequence numbers. Instead, the sequence number for each new connection is incremented by some large number to keep them from being the same. The sequence number may also be incremented by some large number every second (or some other time period). What an external attacker has to do is determine what the values used for these increments are. The attacker can do this by attempting connections at various time intervals in order to observe how the sequence numbers are incremented. Once the pattern is determined, the attacker can attempt a legitimate connection to determine the current value, and then immediately attempt the spoofed connection. The spoofed connection sequence number should be the legitimate connection incremented by the determined value or values.

Sequence numbers are also important in session hijacking, which will be discussed in the "TCP/IP Hijacking" section of this chapter.

Man-in-the-Middle Attacks

A **man-in-the-middle attack**, as the name implies, generally occurs when attackers are able to place themselves in the middle of two other hosts that are communicating in order to view and/or modify the traffic. Ideally, this is done by ensuring that all communication going to or from the target host is routed through the attacker's host (which may be accomplished if the attacker can compromise the router for the target host). The attacker can then observe all traffic before relaying it and can actually modify or block traffic. To the target host, it appears that communication is occurring normally, since all expected replies are received. Figure 15-8 illustrates this type of attack.

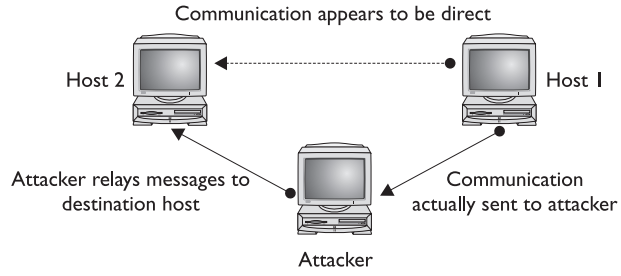

Figure 15-8
A man-in-the-middle attack

The amount of information that can be obtained in a man-in-the-middle attack will obviously be limited if the communication is encrypted. Even in this case, however, sensitive information may still be obtained, since knowing what communication is being conducted, and between which individuals, may in fact provide information that is valuable in certain circumstances.

Man-in-the-Middle Attacks on Encrypted Traffic

The term "man-in-the-middle attack" is sometimes used to refer to a more specific type of attack—one in which the encrypted traffic issue is addressed. Public and private key encryption will be discussed in much greater detail in Chapter 10, but for now it should be understood that public key encryption requires the use of two keys: your public key, which anybody can use to encrypt or "lock" your message, and your private key, which only you know and which is used to "unlock" or decrypt a message locked with your public key.

If you wanted to communicate securely with your friend Bob, you might ask him for his public key so you could encrypt your messages to him. You, in turn, would supply Bob with your public key. An attacker can conduct a man-in-the-middle attack by intercepting your request for Bob's public key and the sending of your public key to him. The attacker would replace your public key with the attacker's public key, and would send it on to Bob. The attacker's public key would also be sent to you by the attacker instead of Bob's public key. Now when either you or Bob encrypt a message, it will be encrypted using the attacker's public key. The attacker can now intercept it, decrypt it, and then send it on by re-encrypting it with the appropriate key for either you or Bob. Each of you thinks you are transmitting messages securely, but in reality your communication has been compromised. Well-designed cryptographic products use techniques such as mutual authentication to avoid this problem.

Replay Attacks

A **replay attack** is exactly what it sounds like: it is an attack where the attacker captures a portion of a communication between two parties and retransmits it at a later time. For example, an attacker might replay a series of commands and codes used in a financial transaction in order to cause the transaction to be conducted multiple times. Generally, replay attacks are associated with attempts to circumvent authentication mechanisms, such as the capturing and reuse of a certificate or ticket.

The best way to prevent replay attacks is with encryption, cryptographic authentication, and time stamps. If a portion of the certificate or ticket includes a date/time stamp or an

expiration date/time, and this portion is also encrypted as part of the ticket or certificate, replaying it at a later time will prove useless, since it will be rejected as having expired.

TCP/IP Hijacking

TCP/IP hijacking and *session hijacking* are terms used to refer to the process of taking control of an already existing session between a client and a server. The advantage to an attacker of hijacking over attempting to penetrate a computer system or network is that the attacker doesn't have to circumvent any authentication mechanisms, since the user has already authenticated and established the session. Once the user has completed the authentication sequence, the attacker can then usurp the session and carry on as if the attacker, and not the user, had authenticated with the system. In order to prevent the user from noticing anything unusual, the attacker may decide to attack the user's system and perform a Denial-of-Service attack on it, taking it down so that the user, and the system, will not notice the extra traffic that is taking place.

Hijack attacks generally are used against web and telnet sessions. The previous discussion on sequence numbers as they applied to spoofing also applies to session hijacking, since the hijacker will need to provide the correct sequence number to continue the appropriated sessions.

Attacks on Encryption

Cryptography is the art of "secret writing," and encryption is the process of transforming *plaintext* into an unreadable format known as *ciphertext* using a specific technique or algorithm. Most encryption techniques use some form of key in the encryption process. The key is used in a mathematical process to scramble the original message to arrive at the unreadable ciphertext. Another key (sometimes the same one and sometimes a different one) is used to decrypt or unscramble the ciphertext to re-create the original plaintext. The length of the key often directly relates to the strength of the encryption.

Cryptanalysis is the process of attempting to break a cryptographic system—it is an attack on the specific method used to encrypt the plaintext. There are various ways cryptographic systems can be compromised. Encryption will be discussed in greater detail in Chapter 10 of this book.

Weak Keys

Certain encryption algorithms may have specific keys that yield poor, or easily decrypted, ciphertext. Imagine an encryption algorithm that consisted solely of a single XOR function (an exclusive OR function where two bits are compared and a 1 is returned if either of the original bits, but not both, is a 1), where the key was repeatedly used to XOR with the plaintext. A key where all bits are 0's, for example, would result in ciphertext that is the same as the original plaintext. This would obviously be a weak key for this encryption algorithm. In fact, any key with long strings of 0's would yield portions of the ciphertext that were the same as the plaintext. In this simple example, there would be many keys that could be considered weak.

Encryption algorithms used in computer systems and networks are much more complicated than a simple, single XOR function, but some algorithms have still been found to have weak keys that make cryptanalysis easier.

Exhaustive Search of Key Space

Even if the specific algorithm used to encrypt a message is complicated and has not been shown to have weak keys, the key length will still play a significant role in how easy it is to attack the method of encryption. Generally speaking, the longer a key is, the harder it will be to attack. Thus, a 40-bit encryption scheme will be easier to attack using a brute-force technique (which tests all possible keys, one by one) than a 256-bit method will be. This is easily demonstrated by imagining a scheme that employed a 2-bit key. Even if the resulting ciphertext were completely unreadable, performing a brute-force attack until one key is found that can decrypt the ciphertext would not take long, since there are only four possible keys. Every bit that is added to the length of a key doubles the number of keys that have to be tested in a brute-force attack on the encryption. It is easy to understand why a scheme utilizing a 40-bit key would be much easier to attack than a scheme that utilized a 256-bit key.

Indirect Attacks

One of the most common ways of attacking an encryption system is to find weaknesses in mechanisms surrounding the cryptography. Examples include poor random number generators, unprotected key exchanges, keys stored on hard drives without sufficient protection, and other general programmatic errors, such as buffer overflows. In attacks that target these types of weaknesses, it is not the cryptographic algorithm itself that is being attacked, but rather the implementation of that algorithm in the real world.

Password Guessing

The most common form of authentication is the user ID and password combination. While it is not inherently a poor mechanism for authentication, the user ID and password combination can be attacked in several ways. All too often, these attacks will yield favorable results for the attacker not as a result of a weakness in the scheme but usually due to the user not following good password procedures.

Poor Password Choices

The least technical of the various password-attack techniques consists of the attacker simply attempting to guess the password of an authorized user of the system or network. It is surprising how often this simple method works, and the reason it does is because people are notorious for picking poor passwords. The problem the users face is that they need to select a password that they can remember. In order to do this, many select simple things, such as their birthday, their mother's maiden name, the name of their spouse or one of their children, or even simply their user ID itself. All it takes is for the attacker to obtain a valid user ID (often a simple matter, because organizations tend to use an individual's names in some combination—first letter of their first name combined with their last name, for example) and a little bit of information about the user before guessing can begin. Organizations sometimes make it even easier for attackers to obtain this sort of information by posting the names of their "management team" and other individuals, sometimes with short biographies, on their web sites.

Even if the person doesn't use some personal detail as their password, the attacker may still get lucky, since many people pick a common word for their password. Attackers can obtain lists of common passwords—there are a number of them on the

Internet. Words such as "password" and "secret" have often been used as passwords. Names of favorite sports teams also often find their way onto lists of commonly used passwords.

Dictionary Attack

Another method of determining passwords is to use a password-cracking program. There are a number of both commercial and public-domain password cracking programs available. The programs use a variety of methods to crack passwords, including using variations on the user ID. These programs often also use a dictionary of words—the words can be used by themselves, or two or more smaller ones may be combined to form a single possible password.

The programs often permit the attacker to create various rules that tell the program how to combine words to form new possible passwords. Users commonly substitute certain numbers for specific letters. If the user wanted to use the word *secret* for a password, for example, the letter *e* may be replaced with the number *3* yielding *s3cr3t*. This password will not be found in the dictionary, so a pure dictionary attack will not crack it. At the same time, the password is still easy for the user to remember. If a rule were created that tried all words in the dictionary and then tried the same words substituting the number *3* for the letter *e*, the password would be cracked.

Rules can also be defined so that the cracking program will substitute special characters for other characters, or combine words together. The ability of the attacker to crack passwords is directly related to the method the user employed to create the password in the first place, as well as the dictionary and rules used.

Brute-Force Attack

If the user has selected a password that will not be found in a dictionary, even if various numbers or special characters are substituted for other letters, the only way the password can be cracked is to attempt a brute-force attack. This entails the password cracking program attempting all possible password combinations.

The length of the password and the size of the set of possible characters in the password will greatly affect the time a brute-force attack will take. A few years ago, this method of attack was very unreliable, since it took considerable time to generate all possible combinations. With the increase in computer speed, however, the time it takes to generate password combinations makes it much more feasible to launch brute-force attacks against certain computer systems and networks. A brute-force attack on a password can take place at two levels. It can be an attack on a system where the attacker is attempting to guess the password at a login prompt, or it can be an attack against the list of passwords contained in a password file. The first attack can be made more difficult by locking the account after a few failed login attempts. The second attack can be thwarted by securely maintaining your password file so that others may not obtain a copy of it.

Birthday Attack

The birthday attack is a special type of brute-force attack. It gets its name from something known as the *birthday paradox*, which states that in a group of at least 23 people, the chance that there will be two individuals with the same birthday is greater than 50 percent. Mathematically, we can use the equation $1.2k^{1/2}$ (with k equaling the size of the

set of possible values), and in the birthday paradox, *k* would be equal to 365 (the number of possible birthdays). This same phenomenon applies to passwords, with *k* just being quite a bit larger.

Software Exploitation

An attack that takes advantage of bugs or weaknesses in software is referred to as **software exploitation**. These weaknesses can be the result of poor design, poor testing, or poor coding practices. They may also result from what are sometimes called "features." An example of this might be a debugging feature, which when used during debugging might allow unauthenticated individuals to execute programs on a system. If this feature is left in when the final version of the software is shipped, it creates a weakness that is just waiting to be exploited.

One common weakness that has been exploited on a number of occasions is buffer overflows. A buffer overflow occurs when a program is provided more data for input than it was designed to handle. For example, what would happen if a program that asks for a 7–10 character phone number instead receives a string of 150 characters? Many programs will provide some error checking to ensure that this will not cause a problem. Some programs, however, do not handle this error, and the extra characters continue to fill memory, overwriting other portions of the program. This can result in a number of problems, including causing the program to abort or the system to crash. Under certain circumstances, the program may execute a command now supplied by the attacker.

Wardialing and WarDriving

Wardialing is the term used to describe an attacker's attempt to discover unprotected modem connections to computer systems and networks. The term's origin comes from the 1983 movie *WarGames,* in which the star has his machine systematically call a sequence of phone numbers in an attempt to find a computer connected to a modem. In the case of the movie, the intent was to find a machine with games the attacker could play, though obviously an attacker could have other purposes once access is obtained.

Wardialing is surprisingly successful, mostly because of rogue modems. These are unauthorized modems attached to computers on a network by authorized users. Generally, the reason for attaching the modem is not malicious—the individual may simply want to be able to go home and then connect to the organization's network in order to continue working. The problem is that if a user can connect, so can an attacker. If the authorized user has not implemented any security protection, access could be totally open. This is often the case. Most organizations have a strict policy against connecting unauthorized modems, but it is hard to enforce. Recently, new technology has been developed to address this common backdoor into corporate networks. Telephone firewalls have been created, which block any unauthorized modem connections into an organization. These devices make it impossible for an unauthorized modem connection to be established and can also enforce strict access policies on any authorized modems.

Another avenue of attack on computer systems and networks that has seen a tremendous increase over the last few years is due to the use of wireless networks. Wireless networks have some obvious advantages—they free employees from the cable connection to a port on their wall, allowing them to wander throughout the building with their machine

and still be connected. An employee could, for example, leave their desk with their laptop and move to a conference room where they could then make a presentation, all without ever having to disconnect their machine from the wall or find a connection in the conference room.

The problem with wireless networks is that it is hard to limit access to them. Since there is no physical connection, the distance that a user can go and still remain connected is a function of the wireless network itself and where the various components of the network are placed. In order to ensure access throughout a facility, stations are often placed at numerous locations, some of which may actually provide access to areas outside of the organization in order to ensure that the farthest offices in the organization can be reached. Frequently, access extends into adjacent offices or even into the parking lot or street. Attackers can locate these access areas that fall outside of the organization and attempt to gain unauthorized access.

The term **WarDriving** has been used to refer to the activity where attackers wander throughout an area (often in a car) toting a computer with wireless capability as they search for wireless networks they can access. There are security measures that can limit an attacker's ability to succeed at this activity, but, just as in wardialing, the individuals who set up the wireless networks don't always activate these security mechanisms.

Social Engineering

Social engineering (covered extensively in Chapter 4) relies on lies and misrepresentation, which an attacker uses to trick an authorized user into providing information or access that the attacker would not normally be entitled to. The attacker might, for example, contact a system administrator pretending to be an authorized user in order to have a password reset. Another common ploy is to pose as a representative from a vendor needing temporary access in order to perform some emergency maintenance. Social engineering also applies to physical access. Simple techniques include impersonating pizza or flower delivery personnel in order to gain physical access to a facility.

Attackers know that, due to poor security practices, if they can gain physical access to an office, the chances are good that, given a little unsupervised time, a user ID and password pair might be found on a notepad or sticky note. Unsupervised access may not even be required, depending on how poor the security practices of the organization are. One of the authors of this book was once considering opening an account at a bank near his home. As he sat down at the desk across from the bank employee taking his information, the author noticed one of the infamous little yellow notes attached to the computer monitor the employee was using. The note read "password for July is julyjuly." It probably isn't too hard to guess what August's password might be. Unfortunately, this is all too often the state of security practices in most organizations. With that in mind, it is easy to see how social engineering might work and might provide all the information needed to gain unauthorized access to a system or network.

Malware

The term **Malware** (also known as *malicious code*) refers to software that has been designed for some nefarious purpose. Such software may be designed to cause damage to a system, such as by deleting all files, or it may be designed to create a backdoor in the

system in order to grant access to unauthorized individuals. Generally, the installation of malicious code is done so that it is not obvious to the authorized users. There are several different types of malicious software, such as viruses, Trojan horse, logic bombs, and worms, and they differ in the ways they are installed and their purposes.

Viruses

The best-known type of malicious code is the **virus**. Much has been written about viruses as a result of several high-profile security events that involved them. A virus is a piece of malicious code that replicates by attaching itself to another piece of executable code. When the other executable code is run, the virus also executes and has the opportunity to infect other files and perform any other nefarious actions it was designed to do. The specific way that a virus infects other files, and the type of files it infects, depends on the type of virus. The first viruses were of two types—boot sector or program viruses.

Boot Sector Virus A boot sector virus infects the boot sector portion of either a floppy disk or a hard drive (just a few years ago, not all computers had hard drives, and many booted from a floppy). When a computer is first turned on, a small portion of the operating system is initially loaded from hardware. This small operating system then attempts to load the rest of the operating system from a specific location (sector) on either the floppy or the hard drive. A boot sector virus infects this portion of the drive.

An example of this type of virus was the Stoned virus, which moved the true Master Boot Record (MBR) from the first to the seventh sector of the first cylinder, and replaced the original MBR with itself. When the system was then turned on, the virus was first executed, which had a one in seven chance of displaying a message stating the computer was "stoned"; otherwise, it would not announce itself and would instead attempt to infect other boot sectors. This virus was rather tame in comparison to other viruses of its time, which often were designed to delete the entire hard drive after a period of time in which they would attempt to spread.

Program Virus A second type of virus is the program virus, which attaches itself to executable files—typically files ending in .exe or .com on Windows-based systems. The virus is attached in such a way that it is executed before the program. Most program viruses also hide a nefarious purpose, such as deleting the hard drive, which is triggered by a specific event, such as a date or after a certain number of other files were infected. Like other types of viruses, program viruses are often not detected until after they execute their malicious payload. One method that has been used to detect this sort of virus before it has an opportunity to damage a system is to calculate checksums for commonly used programs or utilities. Should the checksum for an executable ever change, it is quite likely that this is due to a virus infection.

Macro Virus In the late '90s, another type of virus appeared that now accounts for the majority of viruses. As systems became more powerful, as well as the operating systems that managed them, the boot sector virus, which once accounted for most reported infections, became less common. Systems no longer commonly booted from floppies, which were the main method for boot sector viruses to spread. Instead, the proliferation of software that included macro-programming languages resulted in a new breed of virus—the macro virus.

The Concept virus was the first known example of this new breed. It appeared to be created to demonstrate the possibility of attaching a virus to a document file, something that had been thought to be impossible before the introduction of software that included powerful macro language capabilities. By this time, however, Microsoft Word documents could include segments of code written in a derivative of Visual Basic. Further development of other applications that allowed macro capability, and enhanced versions of the original macro language, had the side effect of allowing the proliferation of viruses that took advantage of this capability.

This type of virus is so common today that it is considered a security best practice to advise users to never open a document attached to an e-mail if it seems at all suspicious. Many organizations now routinely have their mail servers eliminate any attachments containing Visual Basic macros.

Avoiding Virus Infection Always being cautious about executing programs or opening documents given to you is a good security practice. "If you don't know where it came from or where it has been, don't open or run it" should be the basic guideline for all computer users.

Another security best practice for protecting against virus infection is to install and run an antivirus program. Since these programs are designed to protect against known viruses, it is also important to maintain an up-to-date listing of virus signatures for your antivirus software. Antivirus software vendors provide this, and administrators should stay on top of the latest updates to the list of known viruses.

Two advances in virus writing have made it more difficult for antivirus software to detect viruses. These advances are the introduction of *stealth virus* techniques and *polymorphic viruses*. A stealthy virus employs techniques to help evade being detected by antivirus software that uses checksums or other techniques. Polymorphic viruses also attempt to evade detection, but they do so by changing the virus itself (the virus "evolves"). Because the virus changes, signatures for that virus may no longer be valid, and the virus may escape detection by antivirus software.

Virus Hoaxes Viruses have caused so much damage in the last few years that many Internet users have become extremely cautious anytime a rumor of a new virus is heard. Many users will not connect to the Internet when they hear about a virus outbreak, just to be sure they don't get infected themselves. This has given rise to virus hoaxes, in which word is spread about a new virus and the extreme danger it poses. It may warn users to not read certain files or connect to the Internet.

A good example of a virus hoax was the Good Times virus warning, which has been copied repeatedly and can still be seen in various forms today. It caused widespread panic as users read about this extremely dangerous virus, which could actually cause the processor to overheat (from being put into an "nth complexity infinite binary loop") and be destroyed. Many folks saw through this hoax, but many less experienced users did not, and they passed the warning along to all of their friends.

Hoaxes can actually be even more destructive than just wasting time and bandwidth. Some hoaxes warning of a dangerous virus have included instructions to delete certain files if found on the user's system. Unfortunately for those who follow the advice, the files may actually be part of the operating system, and deleting them could keep

the system from booting properly. This suggests another good piece of security advice: make sure of the authenticity and accuracy of any virus report before following somebody's advice. Antivirus software vendors are a good source of factual data for this sort of threat as well (see http://www.symantec.com/avcenter/hoax.html or http://vil.mcafee.com/hoax.asp for examples of hoaxes).

Trojan Horses

A **Trojan horse**, or simply *Trojan*, is a piece of software that appears to do one thing (and may, in fact, actually do that thing) but that hides some other functionality. The analogy to the famous story of antiquity is very accurate. In the original case, the object appeared to be a large wooden horse, and in fact it was. At the same time, it hid something much more sinister and dangerous to the occupants of the city. As long as the horse was left outside the city walls, it could cause no damage to the inhabitants. It had to be taken in by the inhabitants, and it was inside that the hidden purpose was activated. A computer Trojan works in much the same way. Unlike a virus, which reproduces by attaching itself to other files or programs, a Trojan is a stand-alone program that must be copied and installed by the user—it must be "brought inside" the system by an authorized user. The challenge for the attacker is enticing the user to copy and run the program. This generally means that the program must be disguised as something that the user would want to run—a special utility or game, for example. Once it has been copied and is "inside" the system, the Trojan will perform its hidden purpose with the user often still unaware of its true nature.

A good example of a Trojan is Back Orifice (BO), originally created in 1999 and now in several versions. BO can be attached to a number of types of programs. Once it is, and once an infected file is run, BO will create a way for unauthorized individuals to take over the system remotely, as if they were sitting at the console. BO is designed to work with Windows-based systems.

The single best method to prevent the introduction of a Trojan to your system is to never run software if you are unsure of its origin, security, and integrity. A virus-checking program may also be useful in detecting and preventing the installation of known Trojans.

Logic Bombs

Logic bombs, unlike viruses and Trojans, are a type of malicious software that is deliberately installed, generally by an authorized user. A logic bomb is a piece of code that sits dormant for a period of time until some event invokes its often malicious payload. An example of a logic bomb might be a program that is set to automatically load and run, and that periodically checks an organization's payroll or personnel database for a specific employee. If the employee is not found, the malicious payload executes, deleting vital corporate files.

If the trigger is some event, such as not finding a specific name in the personnel file, the code is referred to as a *logic bomb*. If the event is a specific date or time, the program will often be referred to as a *time bomb*. In one famous example of a time bomb, a disgruntled employee left a time bomb in place just prior to being fired from his job. Two weeks later, thousands of client records were deleted. Police were able to eventually track the malicious code to the disgruntled ex-employee, who was prosecuted for his ac-

tions. He had hoped that the two weeks that had passed since his dismissal would have caused investigators to assume he could not have been the individual who had caused the deletion of the records.

Logic bombs are difficult to detect because they are often installed by authorized users and, in particular, have been installed by administrators who are also often responsible for security. This demonstrates the need for a separation of duties and a periodic review of all programs and services that are running. It also illustrates the need to maintain an active backup program so that if your organization loses critical files to this sort of malicious code, you only lose transactions since the most recent backup and don't permanently lose the data.

Worms

Originally, it was easy to distinguish between a **worm** and a virus. Recently, with the introduction of new breeds of sophisticated malicious code, the distinction has blurred. Worms are pieces of code that attempt to penetrate networks and computer systems. Once a penetration occurs, the worm will create a new copy of itself on the penetrated system. Reproduction of a worm thus does not rely on the attachment of the virus to another piece of code or to a file, which is the definition of a virus.

The blurring of the distinction between viruses and worms has come about because of the attachment of malicious code to e-mail. Viruses were generally thought of as a system-based problem, and worms were network-based. If the malicious code is sent throughout a network, it may subsequently be called a worm. The important distinction, however, is whether the code has to attach itself to something else (a virus), or if it can "survive" on its own (a worm).

The Morris Worm

The most famous example of a worm was the Morris worm in 1988. Also sometimes referred to as the Internet worm, because of its effect on the early Internet, the worm was able to insert itself into so many systems connected to the Internet that it has been repeatedly credited with "bringing the Internet to its knees" for several days. It was this worm that provided the impetus for the creation of what was once the Computer Emergency Response Team Coordination Center though is now simply the CERT Coordination Center (CERT/CC) located at Carnegie Mellon University.

The Morris worm was created by a graduate student named Robert Morris. It utilized several known vulnerabilities to gain access to a new system, and it also relied on password guessing to obtain access to accounts. Once a system had been penetrated, a small bootstrap program was inserted into the new system and executed. This program then downloaded the rest of the worm to the new system. The worm had some stealth characteristics to make it harder to determine what it was doing, and it suffered from one major miscalculation. The worm would not be loaded if a copy of it was already found on the new system, but it was designed to periodically ignore this check, reportedly to ensure that the worm could not be easily eliminated. The problem with this plan was that interconnected systems were constantly being reinfected. Eventually, the systems were running so many copies of the worm that the system response time ground to a stop. It took a concerted effort by many individuals before the worm was eliminated. While the Morris worm carried no malicious payload, it is entirely possible for worms to do so.

Code-Red

Much was learned from the Morris Worm, and many new safeguards and procedures were implemented to try and keep from having a similar incident occur. Today, however, the level of connectivity far surpasses what constituted as the Internet in Morris' day, and the sophistication of the average user has also decreased. Coupled with what seems to be a constant discovery of new vulnerabilities in operating systems and application programs, the current environment still provides a breeding ground for new virus and worm outbreaks. With all of our virus protection software, and with the introduction of technology such as firewalls and intrusion detection systems since Morris' time, one might wonder if a new outbreak of a worm could have the same devastating impact on the Internet such as that seen in the Morris incident. The outbreak of the Code-Red worm in July, 2001 effectively answered this question.

On July 19, 2001, over 350,000 computers connected to the Internet were infected by the Code-Red worm. This infection took only 14 hours to occur. The cost estimate for how much damage the worm caused (including variations of the worm released at later dates) exceeds $2.5 billion. There are a couple of interesting points to consider when examining the Code-Red worm, most of which are just as applicable today as they were in 2001 when the worm hit. First of all, the vulnerability exploited by the Code-Red worm was not revealed as a result of the attack. The vulnerability it exploited had been known for a month. The worm took advantage of a buffer-overflow condition in Microsoft's IIS web servers. Microsoft released a patch for this vulnerability along with an announcement of the problem itself on June 18, 2001, and a little less than a month later, on July 12, the first version of the Code-Red worm was released, which took advantage of this vulnerability. The initial version of the worm did little damage as it spread slowly across the Internet, but the worm itself is memory-resident, so simply turning off the computer would not eliminate the worm. Unfortunately, unless the system was patched, a user was likely to become reinfected soon after rebooting and reconnecting to the Internet.

The worm itself didn't carry a malicious payload designed to destroy data on the infected system. On some systems, the message "Hacked by Chinese" was added to the top-level page for the infected host's web site. If the date was between the 1st and 19th of the month, upon infecting a system the worm would generate a random list of IP addresses and attempt to infect them. The first version of the worm used the same seed for the random number generator so each system actually generated the same list of random IP addresses. Thus, this first version of the worm spent much time attempting to reinfect the same machines and, as a consequence, spread very slowly. If the date was between the 20th and the 28th of the month, the worm attempted to launch a Denial-of-Service attack against a web site owned by the White House. After the 28th, the worm remained dormant until the 1st of the following month. All of this changed when the second version of the worm was released on July 19th.

The only change between the first and second versions of the worm was a fix to the seed for the random number generator. The second version of the worm did not use the same seed for each infection. Instead, new seeds were utilized, thus causing a different list of random IP addresses to be created. This small change had a devastating affect on the Internet. No longer were infected systems trying to reinfect the same list of systems. Now each new virus was attempting to infect their own list of IP addresses, resulting in a much wider coverage of the Internet. This resulted in the more than 350,000 machines

becoming infected in a 14-hour period on July 19th. Additional problems were seen with this second version since numerous routers, switches, and other networked devices not susceptible to infection were sometimes still unable to handle the data being sent to them, causing many to crash or reboot. Additional versions of the worm were discovered the next month. The date the second version was released played a large part in keeping the number of infected machines from being even greater. Since the worm was released on the 19th, when the date changed to the 20th, the machines stopped trying to infect other machines and instead started to launch their DoS attack. This provided a period of several days for systems to be rebooted and patched before the worms attempted to infect systems again. This programmed lull played a major part in being able to address this worm.

Slammer

The rapid spread of the Code-Red worm startled many users of the Internet who didn't realize a worm could spread so quickly. If the speed of the Code-Red infection surprised some people, the speed that another worm spread totally amazed everybody else. On Saturday, January 25, 2003, the Slammer virus was released. It exploited a buffer overflow vulnerability in computers running Microsoft's SQL Server or Microsoft SQL Server Desktop Engine. Like the vulnerability in Code-Red, this vulnerability was not new and in fact had been discovered in July, 2002; Microsoft had released a patch for the vulnerability before it was even announced. By the next day, the worm had infected at least 120,000 hosts, had caused network outages, and disrupted airline flights, elections, and ATM services. At its peak, Slammer-infected hosts were generating a reported 1 terabit of worm-related traffic EVERY SECOND.

The truly amazing and frightening aspect of the Slammer worm was its rate of infection. The worm doubled its number of infected hosts every 8 seconds. It is estimated that it took less than 10 minutes to reach global proportions and infect 90 percent of the possible hosts it could infect. This was a dynamic difference when compared to the Code-Red rate, which doubled its number of infected hosts about every 35 minutes. Once a machine was infected (and it took a single packet to infect a machine), the host would start randomly selecting targets and sending packets to them to attempt infection at a rate of 25,000 per second. Like Code-Red, Slammer did not contain a malicious payload. The problems it caused were as a result of the massively overloaded networks, which could not sustain the traffic generated by the thousands of infected hosts. It is interesting to note that this damage was caused with only one out of every thousand systems being vulnerable. One can only imagine the damage and even greater rates of infection had the worm exploited a vulnerability that affected a larger percentage of hosts.

The worm sent its single packet to a specific UDP port—1434—which provided an immediate fix for networks. The response to Slammer was rapid (though not as rapid as the infection itself). Administrators quickly blocked all traffic to UDP port 1434, effectively blocking the spread to new machines. Once again, we can only imagine what the effect would have been had the worm been spread using a more common, and Internet-essential, port such as TCP port 80 (used for HTTP traffic).

Protection Against Worms

How you protect a system against worms depends on the type of worm. Those attached and propagated through e-mail can be avoided by following the same guidelines about

not opening files and not running attachments unless you are absolutely sure of their origin and integrity. Protecting against the Morris type of Internet worm involves securing systems and networks against penetration in the same way you would protect your systems against human attackers. Install patches, eliminate unused and unnecessary services, enforce good password security, and utilize firewalls and intrusion detection systems.

Mobile Code

Mobile code is segments of code sent from another host that is executed on your system. The commonest example of mobile code is web applets written in JAVA. The possible problem with mobile code should be apparent: Can you trust code sent by another system to run on your system? How do you know what the code will do, and that it will only do what the user/system you received it from claims it will do. The threats to your system from hostile mobile code include disclosure of information and modification or damage to information. Safe execution of mobile code will require controlled access to resources. In an ideal environment, this access should really be negotiated for each piece of mobile code received.

Auditing

Auditing, in the financial community, is done to verify the accuracy and integrity of financial records. There have been many standards established in the financial community about how to correctly record and report a company's financial status. In the computer security world, auditing serves a similar function. It is a process of assessing the security state of an organization compared against an established standard. Ensuring that employees are following established procedures and guidelines is one of the methods we can use to address the numerous possible attacks that have been discussed in this chapter.

The important element in auditing is the standard used to evaluate personnel and procedures. Organizations from different communities may have widely different standards, and any audit will need to consider the appropriate elements for the specific community. Audits differ from security or vulnerability assessments in that assessments measure the security posture of the organization but may do so without any mandated standards to compare them against. In a security assessment, generally agreed-upon security "best practices" may be used, but they may lack the regulatory teeth that standards often provide. Penetration tests may also be encountered—these are tests conducted against an organization to see if any holes in the organization's security can be found. The goal of the penetration test is just that, to penetrate the security rather than measuring it against some standard. Penetration tests are often viewed as *white-hat hacking* in that the methods used often mirror those that an attacker might use.

It is important to conduct some form of security audit or assessment on a regular basis. Your organization may spend quite a bit on security, and it is important to measure how effective the efforts have been. In certain communities, audits may be regulated on a periodic basis with very specific standards that must be measured against. Even if your organization is not part of such a community, periodic assessments are important.

There are many things that may be evaluated during an assessment, but at a minimum, the security perimeter (with all of its components, including host-based security)

should be examined, as well as the organization's policies, procedures, and guidelines governing security. Employee training is another aspect that should be examined, since employees are the targets of social engineering and password-guessing attacks.

Security audits, assessments, and penetration tests are a big business, and there are a number of organizations that can perform them. The cost of these varies widely depending on the extent of the test you desire, the background of the company you are contracting with, and the size of the organization to be tested.

Chapter Review

Chapter Summary

After reading this chapter and completing the exercises, you should understand the following regarding malware and attacks on computer systems and networks.

Attacks on Computer Systems and Networks

- Attacks on computer systems and networks can be grouped into two broad categories: attacks on specific software and attacks on a specific protocol or service.
- The target of an attacker can be of two types: targets of opportunity and defined targets. There are a number of different types of attacks, including Denial-of-Service attacks, the installation of backdoors, sniffing, spoofing, man-in-the-middle and replay attacks, TCP/IP session hijacking, wardialing and WarDriving, and attacks on encryption.
- In a Denial-of-Service attack, the attacker is attempting to deny authorized users access either to specific information or to the computer system or network itself.
- The term backdoor is commonly used to refer to programs that attackers install after gaining unauthorized access to a system to ensure that they can continue having unrestricted access to the system.
- Network sniffers are software or hardware devices used to observe traffic as it passes through a network on shared broadcast media.
- Spoofing is a type of attack in which data is made to look like it has come from a different source.
- A man-in-the-middle attack is a type of attack that generally occurs when attackers are able to place themselves in the middle of two other hosts that are communicating, thus allowing the attacker to view and/or modify the traffic.
- Replay attacks are attacks in which the attacker captures a portion of network traffic between two parties and retransmits it at a later time.
- TCP/IP hijacking, also called *session hijacking,* refers to attacks designed to take control of an already existing session between a client and a server.

- All too often attempts at password guessing yield favorable results for the attacker, not as a result of a weakness in the scheme, but usually because the user is not following good password procedures.

Malware

- Malware, also known as *malicious code*, refers to software that has been designed for some nefarious purpose. Malware includes viruses, worms, Trojan horses, logic bombs, and hostile mobile code.
- A virus is a piece of malicious code that replicates by attaching itself to another piece of executable code.
- A Trojan horse, or simply *Trojan*, is a piece of software that appears to do one thing (and may, in fact, actually do that thing) but which hides some other functionality.
- Logic bombs, unlike viruses and Trojans, are a type of malicious software that is deliberately installed, generally by an authorized user. A logic bomb is a piece of code that sits dormant for a period of time until some event invokes its payload.
- Worms are pieces of code that attempt to propagate through penetration of networks and computer systems.

Key Terms

Backdoor *(398)*
Denial-of-Service attack *(396)*
Logic bomb *(412)*
Malware *(409)*
Man-in-the-middle attack *(403)*
Replay attack *(404)*
Sniffers *(399)*
Software exploitation *(408)*
Spoofing *(400)*
TCP/IP hijacking *(405)*
Trojan horse *(412)*
Virus *(410)*
Wardialing *(408)*
WarDriving *(409)*
Worm *(413)*

Key Terms Quiz

Use terms from the Key Terms list to complete the sentences that follow. Don't use the same term more than once. Not all terms will be used.

1. A piece of code that sits dormant for a period of time until some event invokes its payload is known as a _____.
2. A _____ is a software or hardware device used to observe traffic as it passes through a network on shared broadcast media.
3. An attack that takes advantage of bugs or weaknesses in software is called _____.
4. _____ is a type of attack in which data is made to look like it has come from a different source.
5. _____ the term used to describe an attacker's attempt to discover unprotected modem connections to computer systems and networks.
6. An attack in which the attacker is attempting to deny authorized users access either to specific information or to the computer system or network itself is known as a _____ attack.
7. A _____ is a piece of code that attempts to propagate through penetration of networks and computer systems.
8. Attacks designed to take control of an already existing session between a client and a server are called _____ or _____.
9. _____ is the term commonly used to refer to programs that attackers install after gaining unauthorized access to a system to ensure that they can continue to have unrestricted access to that system.
10. A _____ is a piece of malicious code that replicates by attaching itself to another piece of executable code.

Multiple-Choice Quiz

1. A SYN flood is an example of what type of attack?
   - **A.** Malicious code
   - **B.** Denial-of-Service
   - **C.** Man-in-the-middle
   - **D.** Spoofing
2. An attack in which attackers place themselves in the middle of two other hosts that are communicating in order to view and/or modify the traffic is known as…
   - **A.** A man-in-the-middle attack
   - **B.** A Denial-of-Service attack
   - **C.** A sniffing attack
   - **D.** A backdoor attack

3. Which attack takes advantage of a trusted relationship that exists between two systems?
   - **A.** Spoofing
   - **B.** Password guessing
   - **C.** Sniffing
   - **D.** Brute force
4. In what type of attack does an attacker resend the series of commands and codes used in a financial transaction in order to cause the transaction to be conducted multiple times?
   - **A.** Spoofing
   - **B.** Man-in-the-middle
   - **C.** Replay
   - **D.** Backdoor
5. The trick in both spoofing and TCP/IP hijacking is in trying to…
   - **A.** Provide the correct authentication token
   - **B.** Find two systems between which a trusted relationship exists
   - **C.** Guess a password or brute force a password to gain initial access to the system or network
   - **D.** Maintain the correct sequence numbers for the response packets
6. The most ominous aspect of the Slammer worm was the fact that…
   - **A.** It exploited a vulnerability that had previously not been known
   - **B.** It spread so quickly, affecting 90 percent of vulnerable systems in less than 10 minutes
   - **C.** It affected multiple platforms and multiple operating systems as it utilized a segment of mobile code
   - **D.** It contained a malicious payload that was not detected until over a week after the initial attack
7. The ability of an attacker to crack passwords is directly related to the method the user employed to create the password in the first place, as well as…
   - **A.** The length of the password
   - **B.** The size of the character set used in generating the password
   - **C.** The speed of the machine cracking the password
   - **D.** The dictionary and rules used by the cracking program
8. A piece of malicious code that must attach itself to another file in order to replicate is known as…

A. A worm

B. A virus

C. A logic bomb

D. A Trojan

9. A piece of code that attempts to propagate through penetration of networks and computer systems is known as…

A. A worm

B. A virus

C. A logic bomb

D. A Trojan

10. An attack in which the attackers attempt to lie and misrepresent themselves in order to gain access to information that can be useful in an attack is known as what?

A. Social science

B. White-hat hacking

C. Social engineering

D. Social manipulation

11. A virus that attempts to avoid detection by periodically modifying portions of itself would be what type of virus?

A. A stealth virus

B. A delayed propagation virus

C. A polymorphic virus

D. An evasive code virus

12. The best way to minimize possible avenues of attack for your system is to…

A. Install a firewall and check the logs daily

B. Monitor your intrusion detection system for possible attacks

C. Limit the information that can be obtained on your organization and the services that are run by your Internet-visible systems

D. Ensure that all patches have been applied for the services that are offered by your system

13. A wardialing attack is an attempt to exploit what technology?

A. Fiber-optic networks whose cables often run along roads and bridges

B. Cellular telephones

C. Modems and public switched telephone networks (PSTNs)

D. Wireless networks

14. How can you protect against worms of the type that Robert Morris unleashed on the Internet?
    A. Follow the same procedures as you would to secure your system from a human attacker
    B. Install antivirus software
    C. Ensure that no executable attachments to e-mails are executed unless their integrity has been verified
    D. Monitor for changes to utilities and other system software
15. Malicious code that is set to execute its payload on a specific date or at a specific time is known as…
    A. A logic bomb
    B. A Trojan horse
    C. A virus
    D. A time bomb

Essay Quiz

1. Outline the differences between viruses and worms?
2. What is social engineering? Describe how it might be used to gain unauthorized access to a computer system or network.
3. What are the implications of the speed at which the Slammer virus spread?
4. Describe the type of information that attackers might try to obtain if they were able to install a sniffer on a network.
5. Outline the steps in spoofing a system across the Internet.

Lab Projects

Lab Project 15.1

If allowed to do so in your lab, download and install a sniffer and observe the traffic that can be seen on your network. Access a web site and observe the traffic that is sent from your system to the web server.

Lab Project 15.2

E-mail spoofing is a simple matter if you can telnet to port 25 on a system running sendmail. If allowed in your lab, attempt to access this port and send a message to yourself or a friend that is obviously spoofed. Make sure that the "acceptable use policy" for your school and other local rules will allow you to perform this lab before you attempt it.