CHAPTER 13

Types of Attacks and Malicious Software

In this chapter, you will

- Learn about various types of computer and network attacks, including denial-of-service, spoofing, hijacking, and password guessing
- Understand the different types of malicious software that exist, including viruses, worms, Trojan horses, logic bombs, and time bombs
- Explore how social engineering can be used as a means to gain access to computers and networks
- Discover the importance of auditing and what should be audited

Attacks can be made against virtually any layer or level of software, from network protocols to applications. When an attacker finds a vulnerability in a system, he exploits the weakness to attack the system. The effect of an attack depends on the attacker's intent and can result in a wide range of effects, from minor to severe. An attack on a system might not be visible on that system because the attack is actually occurring on a different system, and the data the attacker will manipulate on the second system is obtained by attacking the first system.

Avenues of Attack

A computer system is attacked for two general reasons: it is specifically targeted by the attacker, or it is a target of opportunity. In the first case, the attacker has chosen the target not because of the hardware or software the organization is running but for another reason, such as a political reason. For example, an individual in one country might attack a government system in another country to gather secret information. Or the attacker might target an organization as part of a "hacktivist" attack—the attacker could deface the web site of a company that sells fur coats because the attacker believes using animals in this way is unethical, for example. Perpetrating some sort of electronic fraud is another reason a specific system might be targeted for attack. Whatever the reason, an attack of this nature is usually begun before the hardware and software of the organization are known.

The second type of attack, an attack against a target of opportunity, is launched against a site that has hardware or software that is vulnerable to a specific exploit. The attacker, in this case, is not targeting the organization; he has instead learned of a specific vulnerability and is simply looking for an organization with this vulnerability that he can exploit. This is not to say that an attacker might not be targeting a given sector and looking for a target of opportunity in that sector. For example, an attacker may want to obtain credit card or other personal information and can search for any exploitable company that stores credit card information on its system to accomplish the attack.

Targeted attacks are more difficult and take more time and effort than attacks on a target of opportunity. The latter simply relies on the fact that with any piece of widely distributed software, somebody in the organization will not have patched the system as they should have. Defense against attacks begins with elimination of vulnerabilities exploited by the target of opportunity avenue, as they are also used in targeted attacks.

The Steps in an Attack

Attackers are like bank robbers in the sense that they undergo an organized process when performing an attack. The steps an attacker takes in attempting to penetrate a targeted network are similar to those that a security consultant performs during a penetration test. The following outlines the common steps of the hacking process:

1. Reconnaissance (also known as profiling)
2. Scanning
3. Researching vulnerability
4. Performing the attack

Reconnaissance

The attacker can gather as much information about the organization as possible via several means, including studying the organization's own web site, looking for postings on news groups, or consulting resources such as the Securities and Exchange Commission's (SEC's) Filings & Forms (EDGAR) web site (www.sec.gov/edgar.shtml). A number of different financial reports are available through the EDGAR site that can provide information about an organization that can prove useful for an attack, especially for social engineering attacks. The attacker wants information about IP addresses, phone numbers, names of important individuals, and what networks the organization maintains. The attacker can also use tools such as Whois.Net (www.whois.net) to link IP addresses to registrants.

Scanning

The next step begins the technical part of an attack that determines what target systems are available and active. This is often done using a *ping sweep*, which simply sends a ping (an Internet Control Message Protocol echo request) to the target machine. If the machine responds, the attacker knows it is reachable. His next step is often to perform a *port scan* to help identify which ports are open, which indicates which services may be running on the target machine. The program nmap is the de facto standard for ping sweep-

ing and port scanning. Running nmap with the **–sv** option will perform a *banner grab* in an attempt to determine the version of the software behind open ports. An alternative GUI program for Windows is SuperScan (www.snapfiles.com/get/superscan.html).

NOTE Windows XP Service Pack 2 has removed raw socket access (access to sockets that grant access to packets) and this slows down programs attempting to perform fast and massive sweeps. This effect can be somewhat mitigated by issuing the following command prior to starting the scanning/sweep program: **net stop SharedAccess**.

Determining which operating system is running on the target machine, as well as any specific application programs, occurs after the attacker determines which services are available. Various techniques can be used to send specifically formatted packets to the ports on a target system to view the response. This response will often provide clues as to which operating system and specific applications are running on the target system. Then the attacker should have a list of possible target machines, the operating system running on them, and some specific applications or services to target.

Researching Vulnerability

After the hacker has a list of software running on the systems, he will start researching the Internet for vulnerabilities associated with that software. Numerous web sites provide information on vulnerabilities in specific application programs and operating systems. This information is valuable to administrators who need to know what problems exist and how to patch them.

In addition to information about specific vulnerabilities, some sites also provide tools that can be used to exploit the vulnerabilities. An attacker can search for known vulnerabilities and tools to exploit them, download the information and tools, then use them against a site. If the administrator for the targeted system has not installed the correct patch, the attack may be successful; if the patch has been installed, the attacker will move on to the next possible vulnerability. If the administrator has installed all the appropriate patches so that all known vulnerabilities have been addressed, the attacker may have to resort to a brute-force attack, which involves calculating user ID and password combinations. Unfortunately, this type of attack, which could be easily prevented, sometimes proves successful.

Performing the Attack

Now the attacker is ready to execute an attack, which could have many different results—the system could crash, information could be stolen off the system, or a web site could be defaced. Hackers often install a backdoor and build their own user accounts with administrative privileges so that even when you do patch the system, they can still gain access.

This discussion of attack steps is by no means complete. A system can be attacked in many different ways. The driving force behind the type of attack is the attacker's objective; if activism can be accomplished by a web site defacement, he may consider this a sufficient attack. If the objective is more sinister, such as intellectual property theft or identity theft, data theft may be the hacker's object and hence guide his attack.

Minimizing Possible Avenues of Attack

By understanding the steps an attacker can take, you can limit the exposure of your system and minimize the possible avenues an attacker can exploit. Your first step to minimize possible attacks is to ensure that all patches for the operating system and applications are installed. Many security problems, such as viruses and worms, exploit known vulnerabilities for which patches actually exist. These attacks are successful only because administrators have not taken the appropriate actions to protect their systems.

The next step is to limit the services that are running on the system. As mentioned in earlier chapters, limiting the number of services to those that are absolutely necessary provides two safeguards: it limits the possible avenues of attack (the possible services for which a vulnerability may exist and be exploited), and it reduces the number of services the administrator has to worry about patching in the first place.

Another step is to limit public disclosure of private information about your organization and its computing resources. Since the attacker is after this information, don't make it easy to obtain.

Attacking Computer Systems and Networks

Although 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 used to attack computer systems and networks. This chapter addresses many different ways computers and networks are attacked 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.

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) and attacks on a specific protocol or service. Attacks on a specific application or operating system are generally possible because of 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 either to take advantage of a specific feature of the protocol or service or use the protocol or service in a manner for which it was not intended. This section discusses various forms of attacks of which security professionals need to be aware.

Denial-of-Service Attacks

Denial-of-service (DoS) attacks can exploit a known vulnerability in a specific application or operating system, or they can attack features (or weaknesses) in specific protocols or services. In a DoS attack, the attacker attempts to deny authorized users access either to specific information or to the computer system or network itself. This can be accomplished by crashing the system—taking it offline—or by sending so many requests that the machine is overwhelmed.

The purpose of a DoS attack can be simply to prevent access to the target system, or the attack can be used in conjunction with other actions to gain unauthorized access to a computer or network. For example, a *SYN flooding* attack can be used to prevent service to a system temporarily 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 uses the TCP three-way handshake that establishes a connection between two systems. Under normal circumstances, the first system sends a SYN packet to the system with which it wants to communicate. The second system responds 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 13-1.

NOTE A SYN/ACK is actually the SYN packet sent to the first system, combined with an ACK flag 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 never come, as shown in Figure 13-2. 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 do so, because use of the system has been denied to them.

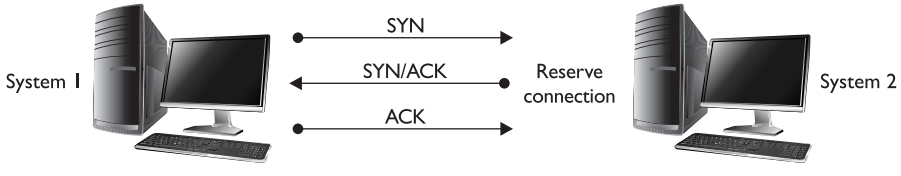

Figure 13-1 The TCP three-way handshake

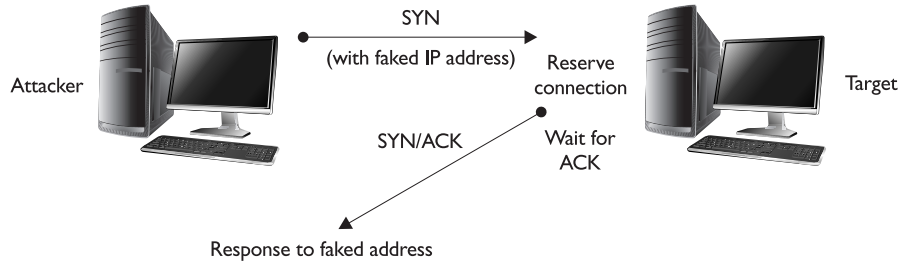

Figure 13-2 A SYN flooding DoS attack

Another simple DoS attack is the infamous 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 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 are not able to handle this size of packet, and the system will hang or crash.

DoS attacks are conducted using a single attacking system. A DoS attack employing multiple attacking systems is known as a distributed denial-of-service (DDoS) attack. The goal of a DDoS attack is also 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, service is denied by overwhelming 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 those targeted in 2000.

Creating a DDoS network is no 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. 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 multi-step process in which the attacker first compromises a few systems that are then used as *handlers* or *masters*, 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 is that with just a few messages to the agents, the attacker can have a flood of messages sent against the targeted system. Figure 13-3 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

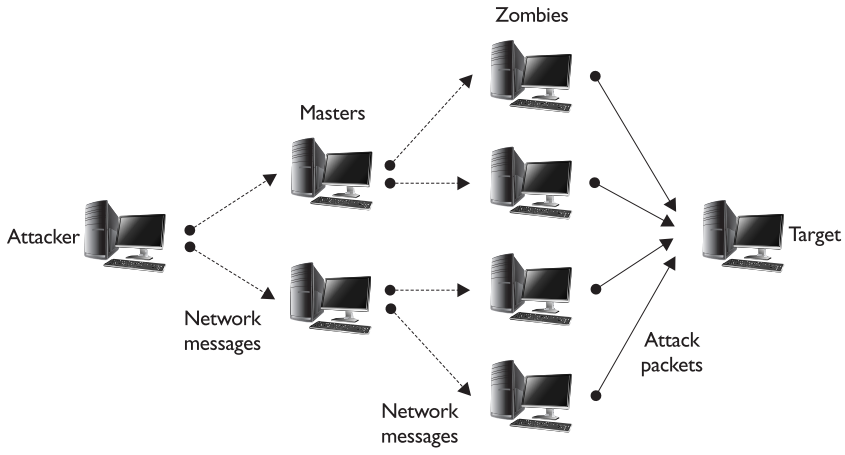

Figure 13-3 DDoS attacks

systems and the applications running on them. Once a specific 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 the vulnerability is discovered and the time exploits become widely available. A vulnerability can also be discovered by hackers, and exploits provide the first clues that a system has been compromised. Attackers can also reverse-engineer patches to learn what vulnerabilities have been patched, allowing them to attack unpatched systems.

Another approach involves changing the time-out option for TCP connections so that attacks such as the SYN flooding attack are more difficult 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 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. Such an approach also involves additional costs to your organization to establish this distributed environment. Addressing the problem in this manner is actually an attempt to mitigate the effect of the attack, rather than preventing or stopping an attack.

To prevent a DDoS attack, you must 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. Many of the current antivirus/spyware security suite tools will detect known zombie-type infections. 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 community of network administrators to test their own systems 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. Carefully consider this approach before implementing it, however, 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 this later in this chapter). Common backdoors include NetBus and Back Orifice. Both of these, if running on your system, can 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.

Null Sessions

Microsoft Windows systems prior to XP and Server 2003 exhibited a vulnerability in their Server Message Block system that allowed users to establish null sessions. A null session is a connection to a Windows interprocess communications share (IPC$). There is good news and bad news associated with this vulnerability. The good news is that Windows XP, Server 2003, and beyond are not susceptible to this vulnerability by default. The bad news is that the millions of previous version machines are vulnerable and patching will not solve the problem. This vulnerability can be used to glean many useful pieces of information from a machine, including user IDs, share names, registry settings, and security settings. A wide range of tools and malware use this vulnerability to achieve their aim.

To harden an affected system from the null session vulnerability requires a bit of work. The seemingly obvious path of upgrading systems to XP and beyond is not a perfect solution, for they too can be tweaked by a malicious user to become susceptible to null sessions. Although there are registry settings to restrict anonymous connections, these will not limit all types; the best method is to limit access to TCP ports 139 and 445 to only trusted users.

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 (looking for logins, for example). 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 13-4. 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, 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).

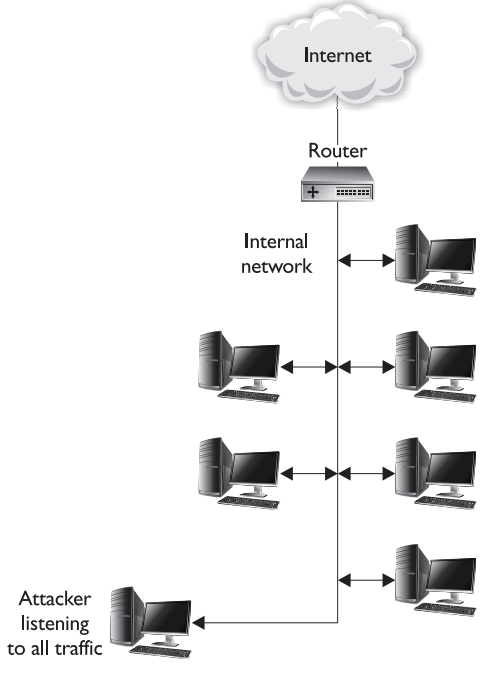

Figure 13-4 Network sniffers listen to all network traffic.

Network sniffers can also be used by attackers to gather information that can be used in penetration attempts. Information such as an authorized 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, 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. This is another reason that physical security is an important part of information security in today's environment.

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 nothing stops you from filling in another system's address. This is one of the several forms of spoofing.

Spoofing E-Mail

In e-mail spoofing, a message is sent with a From address that differs from that of the sending system. This can be easily accomplished in several different ways using several programs. To demonstrate how simple it is to spoof an e-mail address, you can Telnet to port 25 (the port associated with e-mail) on a mail server. 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.

You can use several methods to determine whether 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 originated. A variation on e-mail spoofing, though it is not technically spoofing, is for the attacker to acquire a URL similar to the URL they want to spoof so that e-mail sent from their system appears to have come from the official site—until you read the address carefully. For example, if attackers want to spoof XYZ Corporation, which owns 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 potential viewers that they were at the official site.

IP Address Spoofing

IP is designed to work so that the originators of any IP packet include their own IP address in the From portion of the packet. While this is the intent, nothing 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 can 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 whether 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 13-5. 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.

EXAM TIP A smurf attack allows an attacker to use a network structure to send large volumes of packets to a victim. By sending ICMP requests to a broadcast IP address, with the victim as the source address, the multitudes of replies will flood the victim system.

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

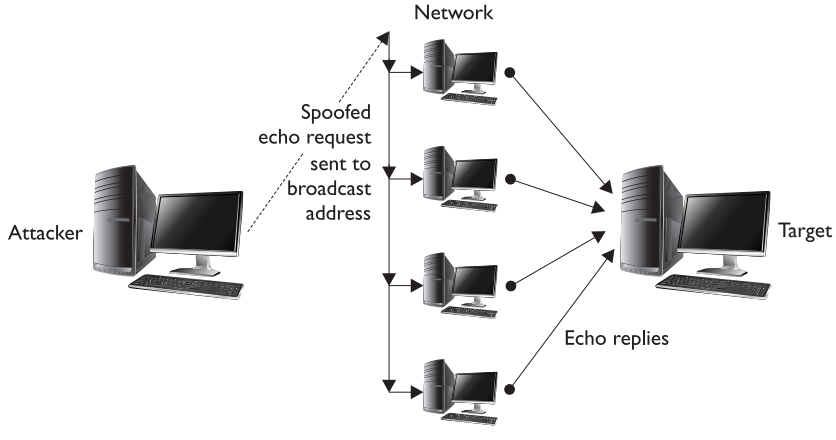

Figure 13-5 Smurfing used in a smurf DoS attack

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 acknowledgement 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 13-6 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 relative 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 a sequence number is 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, two sets of sequence numbers are created, as shown in Figure 13-7. The first system chooses a sequence number to send with the

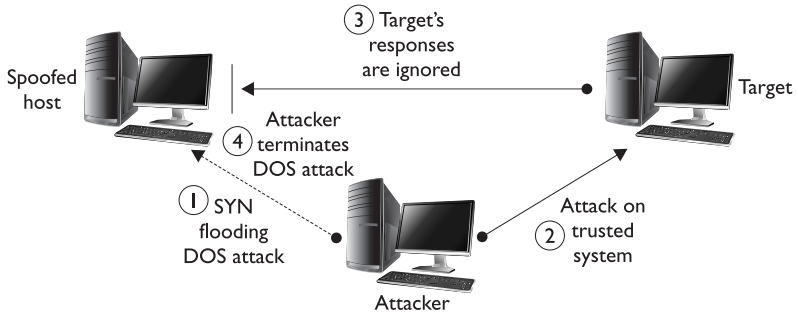

Figure 13-6 Spoofing to take advantage of a trusted relationship

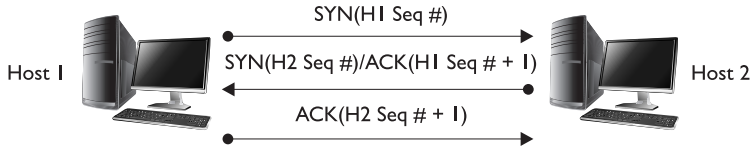

Figure 13-7 Three-way handshake with sequence numbers

original SYN packet. The system receiving this SYN packet acknowledges with a SYN/ACK. It sends an acknowledgement number back, which is based on 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 as the acknowledgement number in the ACK packet with which it responds.

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 with which the target host responds, 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 and the sequence number the target system generates is not observed, it is next to impossible for the attacker to provide the final ACK with the correct sequence number. So the attacker has to guess what the sequence number might be.

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 the numbers from being the same. The sequence number may also be incremented by some large number every second (or some other time period). An external attacker has to determine what values are used for these increments. The attacker can do this by attempting connections at various time intervals 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 is discussed in the "TCP/IP Hijacking" section of this chapter.

Scanning Attacks

Scanners can be used to send specifically crafted packets in an attempt to determine TCP/UDP port status. A XMAS scan, named because the alternating bits in the TCP header look like Christmas lights, uses the URG, PUSH, and FIN flags to determine TCP port availability. If the port is closed, an RST is returned. If the port is open, there is typically no return. A XMAS scan can help determine OS type and version, based upon TCP/IP stack responses, and can also help determine firewall rules. These attacks can also be used to consume system resources, resulting in DoS.

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. Ideally, this is done by ensuring that all communication going to or from the target host is routed through the attacker's host (which can 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 13-8 illustrates this type of attack.

There are numerous methods of instantiating a man-in-the-middle attack; one of the common methods is via session hijacking. Session hijacking can occur when information such as a cookie is stolen, allowing the attacker to impersonate the legitimate session. This attack can be a result of a cross-site scripting attack, which tricks a user into executing code resulting in cookie theft. 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 can 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-key encryption, discussed in detail in Chapter 5, 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 her public key, and she would send this 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.

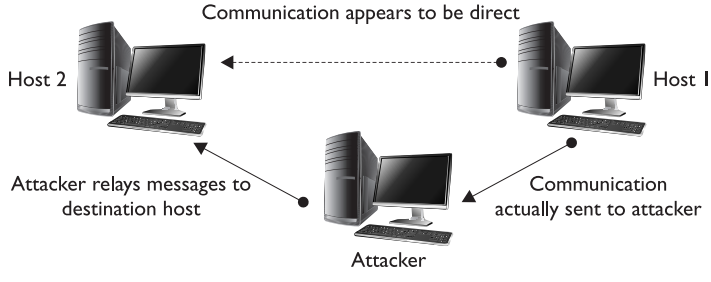

Figure 13-8
A man-in-the-middle attack

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* occurs when 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 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.

EXAM TIP The best method for defending against replay attacks is through the use of encryption and short time frames for legal transactions. Encryption can protect the contents from being understood, and a short time frame for a transaction prevents subsequent use.

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. To prevent the user from noticing anything unusual, the attacker can decide to attack the user's system and perform a DoS 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. Sequence numbers as they apply to spoofing also apply to session hijacking, since the hijacker will need to provide the correct sequence numbers to continue the appropriate 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. Cryptographic systems can be compromised in various ways. Encryption is discussed in detail in Chapter 4.

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 0s, 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 0s would yield portions of the ciphertext that were the same as the plaintext. In this simple example, many keys 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, 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–based scheme. 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 only four keys are possible. 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.

The bottom line is simple: an exhaustive search of the key space will decrypt the message. The strength of the encryption method is related to the sheer size of the key space, which with modern algorithms is large enough to provide significant time constraints when using this method to break an encrypted message. Algorithmic complexity is also an issue with respect to brute force, and you cannot immediately compare different key lengths from different algorithms and assume relative strength.

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.

Address System Attacks

Addresses control many aspects of a computer system. IP addresses can be manipulated, as shown previously, and the other address schemes can be manipulated as well. In the summer of 2008, much was made of a serious domain name system (DNS) vulnerability that required the simultaneous patching of systems by over 80 vendors. This coordinated effort closed a technical loophole in the domain name resolution infrastructure that allowed hijacking and man-in-the-middle attacks on the DNS system worldwide.

The DNS system has been the target of other attacks. One attack, DNS kiting, is an economic attack against the terms of using a new DNS entry. New DNS purchases are allowed a five-day "test period" during which the name can be relinquished for no fee. Creative users learned to register a name, use it for less than five days, relinquish the name, and then get the name and begin all over, repeating this cycle many times using a name without paying for it. Typical registration versus permanent entry ratios of 15:1 occurred in February 2007. GoDaddy reported that out of 55.1 million requests only 3.6 million were not canceled. Another twist on this scheme is the concept of domain name front running, where a registrar places a name on five-day hold after someone searches for it, and then offers it for sale at a higher price. In January 2008, Network Solutions was accused of violating the trust as a registrar by forcing people to purchase names from them after they engaged in domain name tasting.

Another attack on DNS is through the concept of DNS poisoning. DNS poisoning is the unauthorized changing of DNS tables on a machine. When an IP address needs to be resolved, a check against the local cache is performed first. If the address is present, this alleviates the need to ask an outside DNS resource. If the local cache is tampered with, this can result in the hijacking of information as the computer will connect to the wrong site.

Local MAC addresses can also be poisoned in the same manner, although this is called ARP poisoning, which can cause miscommunications locally. Poisoning attacks can be used to steal information, establish man-in-the-middle attacks, and even create DoS opportunities.

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 combination can be attacked in several ways. All too often, these attacks 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. Users need to select a password that they can remember, so they create simple passwords, 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 her password, she may still get lucky, since many people use a common word for their password. Attackers can obtain lists of common passwords—a number of them exist 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 that uses a list of dictionary words to try to guess the password. The words can be used by themselves, or two or more smaller words can be combined to form a single possible password. A number of commercial and public-domain password-cracking programs employ a variety of methods to crack passwords, including using variations on the user ID.

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* could be replaced with the number *3*, yielding *s3cr3t*. This password will not be found in the dictionary, so a pure dictionary attack would not crack it, but 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*, however, 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. The ability of the attacker to crack passwords is directly related to the method the user employs 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 is not found in a dictionary, even if various numbers or special characters are substituted for letters, the only way the password can be cracked is for an attacker to attempt a brute-force attack, in which the password-cracking program attempts 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 time consuming, since it took considerable time to generate all possible combinations. With the increase in computer speed, however, generating password combinations is much faster, making it 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 attack a system where the attacker is attempting to guess the password at a login prompt, or it can attack against the list of password hashes contained in a password file. The first attack can be made more difficult if the account locks after a few failed login attempts. The second attack can be thwarted if the password file is securely maintained so that others cannot obtain a copy of it.

Hybrid Attack

A *hybrid* password attack is a system that combines the preceding methods. Most cracking tools have this option built in, first attempting a dictionary attack, and then moving to brute-force methods.

Birthday Attack

The *birthday attack* is a special type of brute-force attack that gets its name from something known as the *birthday paradox*, which states that in a group of at least 23 people, the chance that two individuals will have the same birthday is greater than 50 percent. Mathematically, we can use the equation $1.25k^{1/2}$ (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 (number of passwords) 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 can 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 remains in the program when the final version of the software is shipped, it creates a weakness that is just waiting to be exploited.

Software exploitation is a preventable problem. Through the use of a secure development lifecycle process, coupled with tools such as threat modeling, bug tracking, fuzzing, and automated code analysis, many of the exploitable elements can be identified and corrected before release. *Fuzzing* is the automated process of applying large sets of inputs to a system and analyzing the output to determine exploitable weaknesses. This technique has been used by hackers to determine exploitable issues and is being adopted by savvy test teams. Identification of potential vulnerabilities by the testing team is the best defense against zero-day attacks, which are attacks against currently unknown vulnerabilities.

Another element that can be exploited is the error messages from an application. Good programming practice includes proper error and exception handling. Proper error handling with respect to the testing team includes the return of significant diagnostic information to enable troubleshooting. Once the code goes to production, the diagnostic information is not as important as it does not help end users, and any potential information that can assist an attacker should be blocked from being presented to the end user. A prime example of this is in SQL injection attacks, where through cleverly crafted injects, a database can be mapped and the data can even be returned to an attacker.

Buffer Overflow Attack

A common weakness that has often been exploited is a *buffer overflow.* 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- to 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, cannot 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 can execute a command supplied by the attacker. Buffer overflows typically inherit the level of privilege enjoyed by the program being exploited. This is why programs that use root level access are so dangerous when exploited with a buffer overflow, as the code that will execute does so at root level access.

Client-Side Attacks

The web browser has become the major application for users to engage resources across the web. The popularity and the utility of this interface has made it a prime target for attackers to gain access and control over a system. A wide variety of attacks can occur via a browser, typically resulting from a failure to properly validate input before use. Unvalidated input can result in a series of injection attacks, header manipulation, and other forms of attack.

Injection Attacks

When user input is used without input validation, this results in an opportunity for an attacker to craft input to create specific events to occur when the input is parsed and used by an application. SQL injection attacks involve the manipulation of input, resulting in a SQL statement that is different than intended by the designer. XML and LDAP injections are done in the same fashion. As SQL, XML, and LDAP are used to store data, this can give an attacker access to data against business rules. Command injection attacks can occur when input is used in a fashion that allows command-line manipulation. This can give an attacker command-line access at the privilege level of the application.

Header Manipulations

When HTTP is being dynamically generated through the use of user inputs, unvalidated inputs can give attackers an opportunity to change HTTP elements. When user-supplied information is used in a header, it is possible to create a variety of attacks, including cache-poisoning, cross-site scripting, cross-user defacement, page hijacking, cookie manipulation, or open redirect.

Malicious Code

Malicious code refers to software that has been designed for some nefarious purpose. Such software can be designed to cause damage to a system, such as by deleting all files, or it can be designed to create a backdoor in the system to grant access to unauthorized individuals. Generally the installation of malicious code is done so that it is not obvious to the authorized users. Several different types of malicious software can be used, such as viruses, Trojan horses, logic bombs, spyware, 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 created 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 (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 the virus code. When the system was 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 were often 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 executes. Most program viruses also hide a nefarious purpose, such as deleting the hard drive data, which is triggered by a specific event, such as a date or after a certain number of other files are 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 it is due to a virus infection.

Macro Virus In the late 1990s, another type of virus appeared that now accounts for the majority of viruses. As systems and operating systems became more powerful, 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 never to 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 sent 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 mantra 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 information, 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 stealth 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 to systems 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 their machines don't get infected. 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 they're 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: ensure 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 www.symantec.com/business/security_response/threatexplorer/risks/hoaxes.jsp 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 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 of Troy. 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 standalone 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 offered in several versions. BO can be attached to a number of types of programs. Once it is attached, 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. Many Trojans communicate to the outside through a port that the Trojan opens, and this is one of the ways Trojans can be detected.

The single best method to prevent the introduction of a Trojan to your system is never to 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.

Spyware

Spyware is software that "spies" on users, recording and reporting on their activities. Typically installed without user knowledge, spyware can perform a wide range of activities. It can record keystrokes (commonly called *keylogging*) when the user logs onto specific web sites. It can monitor how a user applies a specific piece of software, that is, monitor attempts to cheat at games. Many spyware uses seem innocuous at first, but the unauthorized monitoring of a system can be abused very easily. In other cases, the spyware is specifically designed to steal information. Many states have passed legislation banning the unapproved installation of software, but spyware can circumvent this issue through complex and confusing end-user license agreements.

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 malicious payload. An example of a logic bomb might be a program that is set to load and run automatically, 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 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 eventually able to track the malicious code to the disgruntled ex-employee, who was prosecuted for his actions. 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 on a system. 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, it loses only transactions that occurred since the most recent backup and there is no permanent loss of data results.

Rootkits

Rootkits are a form of malware that is specifically designed to modify the operation of the operating system in some fashion to facilitate nonstandard functionality. The history of rootkits goes back to the beginning of the UNIX operating system, where they were sets of modified administrative tools. Originally designed to allow a program to take greater control over operating system function when it fails or becomes unrespon-

sive, the technique has evolved and is used in a variety of ways. One high-profile case occurred at Sony BMG Corporation, when rootkit technology was used to provide copy protection technology on some of the company's CDs. Two major issues led to this being a complete debacle for Sony: first, the software modified systems without the users' approval; and second, the software opened a security hole on Windows-based systems, creating an exploitable vulnerability at the rootkit level. This led the Sony case to be labeled as *malware*, which is the most common use of rootkits.

A rootkit can do many things—in fact, it can do virtually anything that the operating system does. Rootkits modify the operating system kernel and supporting functions, changing the nature of the system's operation. Rootkits are designed to avoid, either by subversion or evasion, the security functions of the operating system to avoid detection. Rootkits act as a form of malware that can change thread priorities to boost an application's performance, perform keylogging, act as a sniffer, hide other files from other applications, or create backdoors in the authentication system. The use of rootkit functionality to hide other processes and files enables an attacker to use a portion of a computer without the user or other applications knowing what is happening. This hides exploit code from antivirus and antispyware programs, acting as a cloak of invisibility.

Rootkits can load before the operating system loads, acting as a virtualization layer, as in SubVirt and Blue Pill. Rootkits can exist in firmware, and these have been demonstrated in both video cards and PCI expansion cards. Rootkits can exist as loadable library modules, effectively changing portions of the operating system outside the kernel. Further information on specific rootkits in the wild can be found at www.antirootkit.com.

EXAM TIP Five types of rootkits exist: firmware, virtual, kernel, library, and application level.

Once a rootkit is detected, it needs to be removed and cleaned up. Because of rootkits' invasive nature, and the fact that many aspects of rootkits are not easily detectable, most system administrators don't even attempt to clean up or remove a rootkit. It is far easier to use a previously captured clean system image and reimage the machine than attempt to determine the depth and breadth of the damage and attempt to fix individual files.

Worms

It was once 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.

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).

Some recent examples of worms that have had high profiles include the Sobig worm of 2003, the SQL Slammer worm of 2003, the 2001 attacks of Code Red and Nimba, and the 2005 Zotob worm that took down CNN Live. Nimba was particularly impressive in that it used five different methods to spread; via e-mail, via open network shares, from browsing infected web sites, using directory traversal vulnerability of Microsoft IIS 4.0/5.0, and most impressively through the use of backdoors left by Code Red II and sadmind worms.

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. This worm provided the impetus for the creation of what was once called the Computer Emergency Response Team Coordination Center, now the CERT Coordination Center (CERT/CC), located at Carnegie Mellon University.

The Morris worm was created by graduate student 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 was 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 ignore this check periodically, 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 to eliminate the worm. While the Morris worm carried no malicious payload, it is entirely possible for worms to do so.

Samy Worm (The MySpace Worm) MySpace is a popular social networking site with a feature that allows people to list other users as friends. In 2005, a clever MySpace user looking to expand his friends list created the first self-propagating cross-site scripting (XSS) worm. In less than a day, the worm had gone viral and user Samy had amassed more than 1 million friends on the popular online community. The worm's code, now posted at http://namb.la/popular/tech.html, used a fairly sophisticated JavaScript. Fortunately the script was written for fun and didn't try to take advantage of unpatched security holes in Internet Explorer to create a massive MySpace botnet. MySpace was taken down as the worm replicated too efficiently, eventually surpassing several thousand replications per second.

Protection Against Worms How you protect your 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. In-

stall patches, eliminate unused and unnecessary services, enforce good password security, and use firewalls and intrusion detection systems. More sophisticated attacks, such as the Samy worm, are almost impossible to avoid.

Application-Level Attacks

Attacks against a system can occur at the network level, at the operating system level, at the application level, or at the user level (social engineering). Early attack patterns were against the network, but most of today's attacks are aimed at the applications. This is primarily because this is where the objective of most attacks resides; in the infamous words of bank robber Willie Sutton, "because that's where the money is." In fact, many of today's attacks on systems combinations of use vulnerabilities in networks, operating systems, and applications, all means to an end to obtain the desired objective of an attack, which is usually some form of data.

Application level attacks take advantage of several facts associated with computer applications. First, most applications are large programs written by groups of programmers, and by their nature have errors in design and coding that create vulnerabilities. For a list of typical vulnerabilities, see the Common Vulnerability and Exposures (CVE) list maintained by Mitre, http://cve.mitre.org. Second, even when vulnerabilities are discovered and patched by software vendors, end users are slow to apply patches, as evidenced by the SQL Slammer incident in January 2003. The vulnerability exploited was a buffer overflow, and the vendor supplied a patch six months prior to the outbreak, yet the worm still spread quickly due to the multitude of unpatched systems. A more complete examination of common application vulnerabilities is presented in Chapter 15.

Secure Software Development Lifecycle

The attacks previously discussed take advantage of vulnerabilities in the software that runs on computers, from the network OS, to the system OS, to the application itself—all have been attacked. Over the past decade, significant advancement has been made in developing processes that reduce the vulnerabilities in software. The majority of these improvements have come from a disciplined process known as *secure software development lifecycle*, a series of additional steps applied to whichever software development methodology is being used. These steps have been shown to reduce the vulnerabilities in the software through better code and code reviews, finding and fixing the errors before the product is released. These steps have reduced serious vulnerabilities by over 80 percent at companies with mature secure development processes.

The elements of a secure development lifecycle process include attack surface analysis, threat modeling, bug tracking, fuzzing, code reviews, and an expanded testing methodology. *Attack surface analysis* is the analysis of code to see how many entry points are in place and whether a reduction in entry points can reduce the number of ways an attacker can attack the code. *Threat modeling* is a communication tool that details how the software can be attacked by an adversary, giving the entire design and development team a chance to see how their design and implementation could be attacked, so that vulnerabilities can be closed or mitigated.

Testing for security is a much broader series of tests than just functional testing. Misuse cases can be formulated to verify that vulnerabilities cannot be exploited. Fuzz testing uses random inputs to check for exploitable buffer overflows. Code reviews by design and development teams are used to verify that security elements such as input and output validation are functional, as these are the best defenses against a wide range of attacks, including cross-site scripting and cross-site request forgeries. Code walkthroughs begin with design reviews, architecture examinations, unit testing, subsystem testing, and ultimately, complete system testing.

Testing includes *white-box testing*, where the test team has access to the design and coding elements; *black-box testing*, where the team does not have access; and *grey-box testing*, where information is greater than black-box testing but short of white-box testing. These different modes of testing are used for different objectives; for example, fuzzing works perfectly fine regardless of the type of testing, whereas certain types of penetration tests are better in a white-box testing environment. Testing is also performed on the production code to verify that error handling and exception reporting, which may provide detailed diagnostic information during development, are squelched to prevent information release during error conditions.

Final code is subjected to *penetration tests*, designed specifically to test configuration, security controls, and common defenses such as input and output validation and error handling. Penetration testing can explore the functionality and whether or not specific security controls can be bypassed. Using the attack surface analysis information, penetration testers can emulate adversaries and attempt a wide range of known attack vectors in order to verify that the known methods of attack are all mitigated.

War-Dialing and War-Driving

War-dialing is the term used to describe an attacker's attempt to discover unprotected modem connections to computer systems and networks. The term's origin is 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.

War-dialing is surprisingly successful, mostly because of *rogue modems*—unauthorized modems attached to computers on a network by authorized users. Generally the reason for attaching the modem is not malicious—an individual may simply want to be able to go home and then connect to the organization's network to continue working. The problem, however, is that if a user can connect, so can an attacker. If the authorized user has not implemented any security protection, this means of access could be totally open. This is often the case. Most organizations enact strict policies against connecting unauthorized modems, but it is difficult to enforce this kind of policy. 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 has seen a tremendous increase over the last few years because of the increase in 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 move throughout the building with their laptops and still be connected. An employee could, for example, leave her desk with her laptop and move to a conference room where she could then make a presentation, all without ever having to disconnect her machine from the wall or find a connection in the conference room.

The problem with wireless networks is that it is difficult to limit access to them. Since no physical connection exists, 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. 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 to ensure that the farthest offices in the organization can be reached. Frequently, access extends into adjacent offices or 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 *war-driving* has been used to refer to the activity in which attackers wander throughout an area (often in a car) with a computer with wireless capability, searching for wireless networks they can access. Some security measures can limit an attacker's ability to succeed at this activity, but, just as in war-dialing, the individuals who set up the wireless networks don't always activate these security mechanisms.

Social Engineering

Social engineering relies on lies and misrepresentation, which an attacker uses to trick an authorized user into providing information or access the attacker would not normally be entitled to. The attacker might, for example, contact a system administrator pretending to be an authorized user, asking to have a password reset. Another common ploy is to pose as a representative from a vendor needing temporary access to perform some emergency maintenance. Social engineering also applies to physical access. Simple techniques include impersonating pizza or flower delivery personnel 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 might not even be required, depending on the quality of the security practices of the organization. (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 an attacker needs to gain unauthorized access to a system or network.

Auditing

Auditing, in the financial community, is done to verify the accuracy and integrity of financial records. Many standards have been established in the financial community about how to record and report a company's financial status correctly. 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.

The important elements here are the standards. 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 against which to compare them. In a security assessment, general security "best practices" can be used, but they may lack the regulatory teeth that standards often provide. Penetration tests can also be encountered—these tests are conducted against an organization to determine whether any holes in the organization's security can be found. The goal of the penetration test is 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 attackers (often called *black hats*) might use.

You should conduct some form of security audit or assessment on a regular basis. Your organization might spend quite a bit on security, and it is important to measure how effective the efforts have been. In certain communities, audits can 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.

Many particulars can 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 studied, since employees are the targets of social engineering and password-guessing attacks.

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

A powerful mechanism for detecting security incidents is the use of security logs. For logs to be effective, they require monitoring. Monitoring of event logs can provide information concerning the events that have been logged. This requires advance decisions about the items to be logged. Logging too many items uses a lot of space and increases the workload for personnel who are assigned the task of reading logs. The same is true for security, access, audit, and application-specific logs. The bottom line is that while logs are valuable, preparation is needed to determine the correct items to be logged and the mechanisms by which logs are reviewed. Security Information Event Management (SIEM) software can assist in the analysis of log files.

Chapter Review

In attempting to attack a computer system or network, an attacker follows several general steps. These include gathering as much information about the target as possible, obtaining information about potential vulnerabilities that might exist in the operating system or applications running on the target system, and finally using tools to attempt to exploit those vulnerabilities. An administrator can make this process more difficult for the attacker by limiting the amount of information that can be obtained about the organization, by limiting the services offered, and by installing all appropriate patches for the remaining services.

Attackers can access computer systems and networks in a number of different ways. These vary from the non-technical social engineering attacks, where attackers attempt to lie and misrepresent themselves to authorized users in order to obtain key information, to DDoS attacks, which can incorporate thousands of penetrated systems in an attack on a targeted system or network.

In addition to guarding against human attackers, you must try to prevent various forms of malicious software from attacking your system. Security auditing and assessments can be used to measure an organization's current security posture. It is important that you understand the various types of attacks that could affect your organization to plan how you will address them, should they occur.

Questions

To further help you prepare for the Security+ exam, and to test your level of preparedness, answer the following questions and then check your answers against the list of correct answers at the end of the chapter.

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 the attacker simply listens for all traffic being transmitted across a network, in the hope of viewing something such as a user ID and password combination, 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 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. Rootkits are challenging security problems because
   - **A.** They can be invisible to the operating system and end user.
   - **B.** Their true functionality can be cloaked, preventing analysis.
   - **C.** They can do virtually anything an operating system can do.
   - **D.** All of the above.
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 to replicate itself is known as
   - **A.** A worm
   - **B.** A virus

C. A logic bomb

D. A Trojan

9. A piece of malicious code that appears to be designed to do one thing (and may in fact do that thing) but that hides some other payload (often malicious) is known as

A. A worm

B. A virus

C. A logic bomb

D. A Trojan

10. An attack in which an attacker attempts to lie and misrepresent himself in order to gain access to information that can be useful in an attack is known as

A. Social science

B. White-hat hacking

C. Social engineering

D. Social manipulation

11. The first step in an attack on a computer system consists of

A. Gathering as much information about the target system as possible

B. Obtaining as much information about the organization in which the target lies as possible

C. Searching for possible exploits that can be used against known vulnerabilities

D. Searching for specific vulnerabilities that may exist in the target's operating system or software applications

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 war-driving attack is an attempt to exploit what technology?

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

B. Cellular telephones

C. The public switched telephone network (PSTN)

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 you'd use 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

Answers

1. **B.** A SYN flood attack involves launching a large number of SYN packets at a system. In TCP, the response to this is a SYN/ACK, and the system then waits for an ACK to complete the three-way handshake. If no ACK is received, the system will wait until a time-out occurs, and then it will release the connection. If enough SYN packets are received (requesting that communication be set up) the system can fill up and not process any more requests. This is a type of DoS attack.
2. **C.** Sniffing consists of a person simply listening to all traffic on a network. It takes advantage of the friendly nature of the network, in which systems are only supposed to grab and examine packets that are destined for them. Sniffing looks at all packets traveling across the network.
3. **A.** One form of spoofing attack attempts to take advantage of the trusted relationship that may exist between two systems. This trusted relationship could mean that users on one system will not be required to authenticate themselves when accessing the other system; the second system trusts the first to have performed any necessary authentication. If packets are formed that claim to have come from one of the trusted systems, the target can be fooled into performing actions as if an authorized user had sent them.
4. **C.** This is the description of a replay attack.
5. **D.** Getting the correct sequence number is the tricky part of any attempt to spoof or take over a session. This is made easy if the attacker can observe (sniff) the network traffic. If, however, the attacker is external to the network, the task is much more complicated.

6. **D.** Rootkits have almost unlimited power over an infected system. They can cloak themselves from detection and hide their true nature.
7. **D.** This is a tricky question. All of the answers have a bearing on the ability of the attacker to crack the password, but, as discussed in the text, the dictionary and rule set used will make or break the attempt (unless an attacker *wants* to try a brute-force attack, which is generally his last option). The size of the password will certainly have a bearing, but the difference between brute-forcing a 13-character password and a 14-character password is not important—neither will be accomplished in the lifetime of the attacker. The same can be said of the size of the character set used to generate the password. The more characters that are available, the larger the number of passwords that must be tried in order to brute-force it—but attackers try to stay away from using brute-force attacks. The speed of the machine will have some bearing, but speed will make little difference if the attacker uses a brute-force attack, since he still won't crack it in time to take advantage of it. If the attacker can pick a good dictionary and rule set, he can probably crack the password (remember that users have a tendency to select poor passwords).
8. **B.** This answer defines a virus. This is the distinguishing aspect of a virus that separates it from other forms of malicious code, especially worms.
9. **D.** This describes a Trojan (or Trojan horse). A virus that is attached to another file and that appears to be that file may also hide a malicious payload, but the description provided is traditionally used to describe a Trojan.
10. **C.** This is a description of social engineering. The term *white-hat hacking* is often used to refer to authorized penetration tests on a network.
11. **B.** The first step is generally acknowledged to be to gather as much information about the organization as possible. This information can then be used in social engineering attacks that can result in the revelation of even more information, or even access to the system. If access can be obtained without having to run any exploits, the attacker's chance of discovery is minimized. The second step is to gather information about the specific systems and networks—details on the actual hardware and software that is being used. It is not until both of these steps have been accomplished that possible vulnerabilities and tools to exploit them can be determined. This sequence may differ if the attacker is not targeting a specific system, but is instead looking for systems that are vulnerable to a specific exploit. In this case, the attacker would probably be searching for a vulnerability first, and then for a tool that exploits it, and he may never even consider the organization that is being targeted.
12. **C.** To minimize the avenues of attack, you need to limit the information that can be obtained and the number of services you offer. The more services that are available, the greater the number of possible avenues that can be exploited. It is important to install patches, but this doesn't minimize the

avenues; it protects specific avenues from attack. The use of firewalls and intrusion detection systems is important, but monitoring them doesn't aid in minimizing the avenues of attack (though a properly administered firewall can help to limit the exposure of your network).

13. **D.** War-driving is an attempt to locate wireless networks whose access area extends into publicly accessible space.
14. **A.** The Morris worm used the same type of techniques to penetrate the systems that human attackers use. Therefore, if you protect the system against one, you are protecting it against the other. Installing an antivirus package and not allowing executable attachments to e-mail to be executed are good ideas, but they address the other type of worm, not the Morris type of Internet worm. Monitoring the system for changes to utilities and other system software is also a good idea, but it is reactive in nature and discovering these changes means the individual or worm has already penetrated your system. Your goal should be to try to prevent this in the first place.
15. **D.** This defines a time bomb. The more general term *logic bomb* is sometimes used, but this term generally refers to a piece of software that is set to execute when some specified event occurs. When that event is a date or time, we often refer to the malicious code as a time bomb.