Chapter
1

Threats, Attacks, and Vulnerabilities

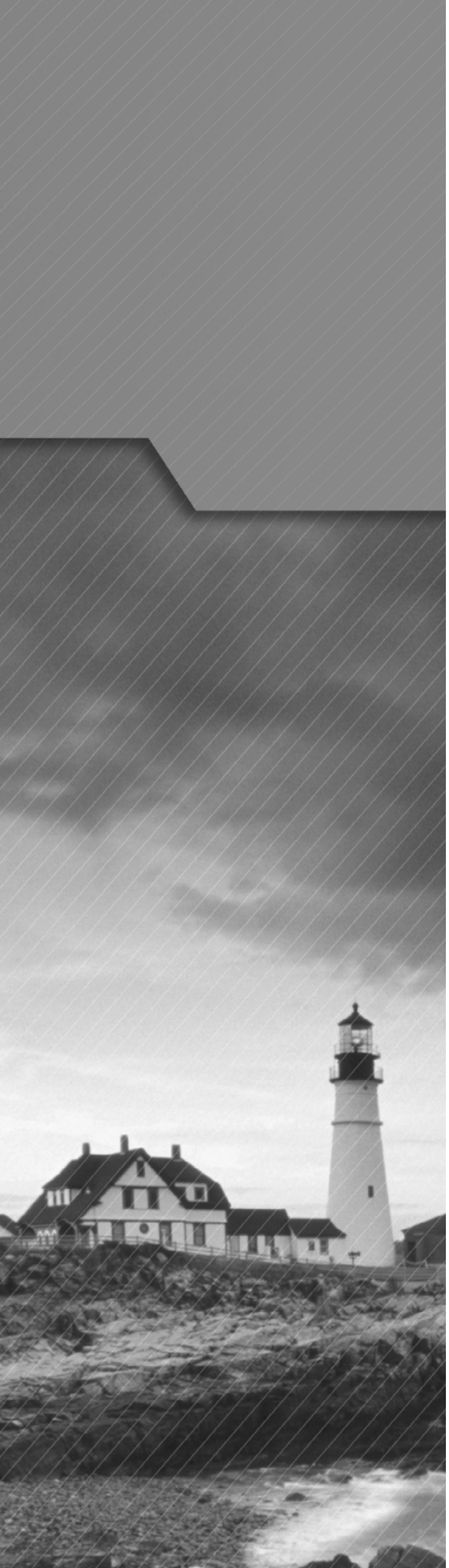
COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE THE FOLLOWING:

✓ **1.1 Given a scenario, analyze indicators of compromise and determine the type of malware.**

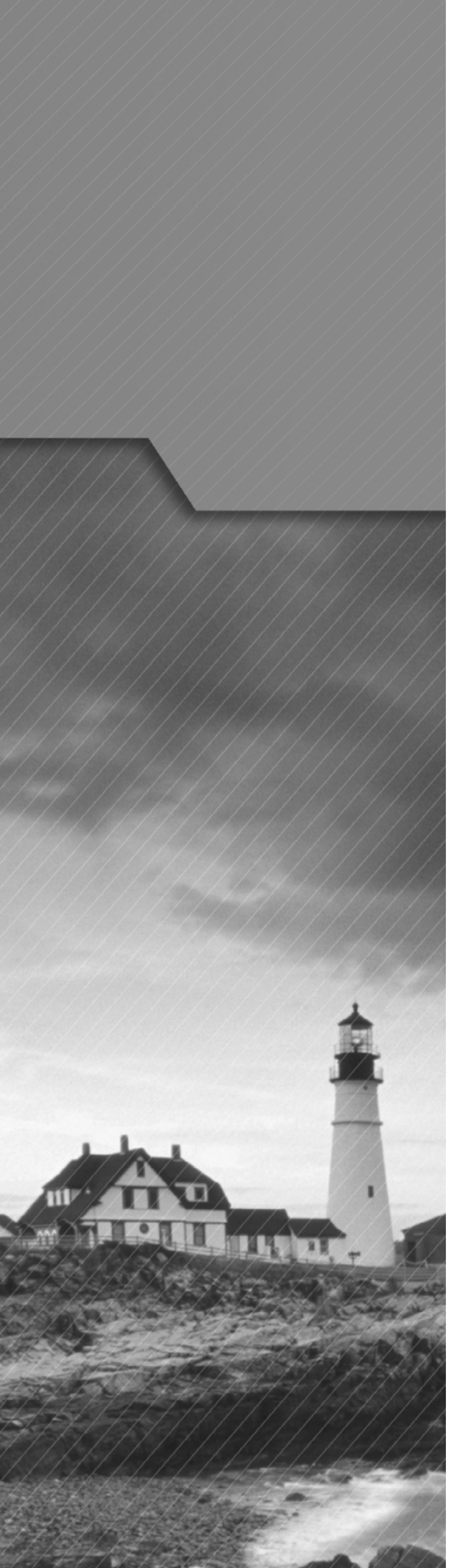
- Viruses
- Crypto-malware
- Ransomware
- Worm
- Trojan
- Rootkit
- Keylogger
- Adware
- Spyware
- Bots
- RAT
- Logic bomb
- Backdoor

✓ **1.2 Compare and contrast types of attacks.**

- Social engineering
  - Phishing
  - Spear phishing
  - Whaling
  - Vishing
  - Tailgating

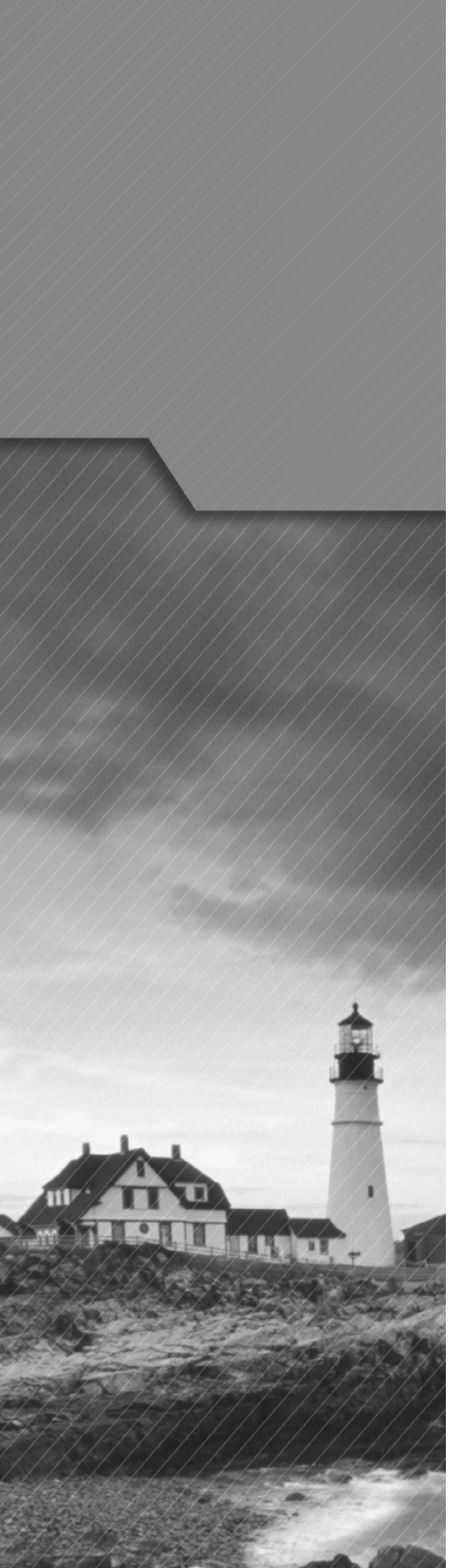


    - Impersonation
    - Dumpster diving
    - Shoulder surfing
    - Hoax
    - Watering hole attack
    - Principles (reasons for effectiveness)
        - Authority
        - Intimidation
        - Consensus
        - Scarcity
        - Familiarity
        - Trust
        - Urgency
- Application/service attacks
    - DoS
    - DDoS
    - Man-in-the-middle
    - Buffer overflow
    - Injection
    - Cross-site scripting
    - Cross-site request forgery
    - Privilege escalation
    - ARP poisoning
    - Amplification
    - DNS poisoning
    - Domain hijacking
    - Man-in-the-browser
    - Zero day
    - Replay
    - Pass the hash



- Hijacking and related attacks
  - Clickjacking
  - Session hijacking
  - URL hijacking
  - Typo squatting
- Driver manipulation
  - Shimming
  - Refactoring
- MAC spoofing
- IP spoofing

- Wireless attacks
  - Replay
  - IV
  - Evil twin
  - Rogue AP
  - Jamming
  - WPS
  - Bluejacking
  - Bluesnarfing
  - RFID
  - NFC
  - Disassociation
- Cryptographic attacks
  - Birthday
  - Known plain text/cipher text
  - Rainbow tables
  - Dictionary
  - Brute force
    - Online vs. offline
  - Collision
  - Downgrade



- Replay
- Weak implementations

✓ **1.3 Explain threat actor types and attributes.**

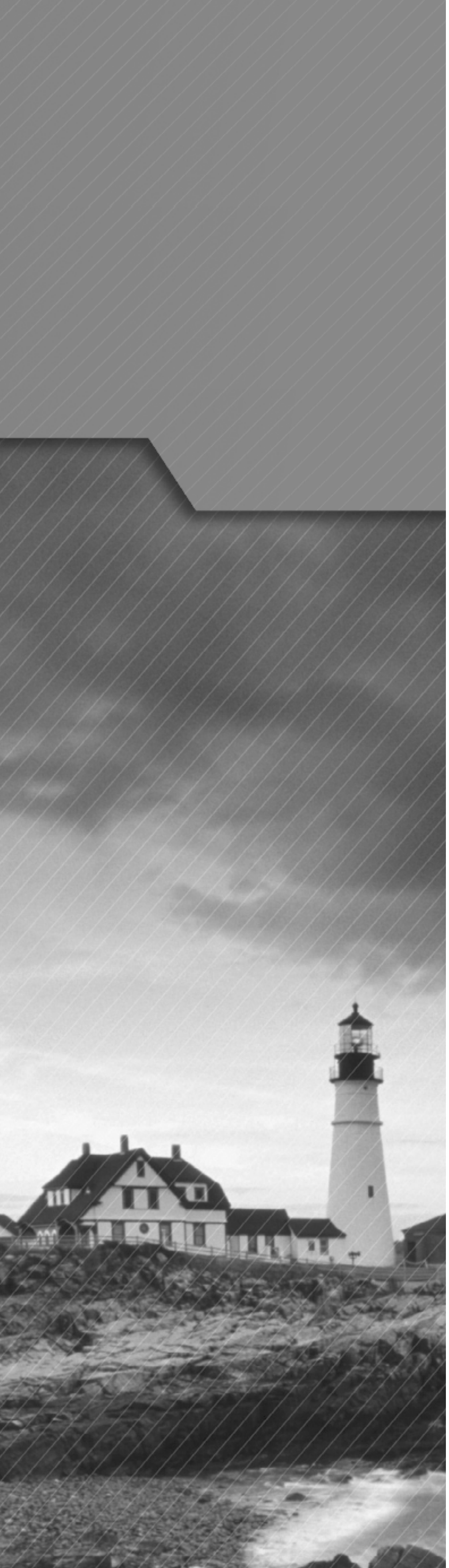
- Types of actors
  - Script kiddies
  - Hacktivist
  - Organized crime
  - Nation states/APT
  - Insiders
  - Competitors
- Attributes of actors
  - Internal/external
  - Level of sophistication
  - Resources/funding
  - Intent/motivation
- Use of open-source intelligence

✓ **1.4 Explain penetration testing concepts.**

- Active reconnaissance
- Passive reconnaissance
- Pivot
- Initial exploitation
- Persistence
- Escalation of privilege
- Black box
- White box
- Gray box
- Pen testing vs. vulnerability scanning

✓ **1.5 Explain vulnerability scanning concepts.**

- Passively test security controls
- Identify vulnerability



- Identify lack of security controls
- Identify common misconfigurations
- Intrusive vs. non-intrusive
- Credentialed vs. non-credentialed
- False positive

✓ 1.6 Explain the impact associated with types of vulnerabilities.

- Race conditions
- Vulnerabilities due to:
  - End-of-life systems
  - Embedded systems
  - Lack of vendor support
- Improper input handling
- Improper error handling
- Misconfiguration/weak configuration
- Default configuration
- Resource exhaustion
- Untrained users
- Improperly configured accounts
- Vulnerable business processes
- Weak cipher suites and implementations
- Memory/buffer vulnerability
  - Memory leak
  - Integer overflow
  - Buffer overflow
  - Pointer dereference
  - DLL injection
- System sprawl/undocumented assets
- Architecture/design weaknesses
- New threats/zero day
- Improper certificate and key management

The Security+ exam will test your knowledge of IT attacks and compromises. There are a wide range of hacks and compromises that both individuals and organizations must understand in order to defend against downtime and intrusion. To pass the test and be effective in reducing loss and harm, you need to understand the threats, attacks, vulnerabilities, concepts, and terminology detailed in this chapter.

1.1 Given a scenario, analyze indicators of compromise and determine the type of malware.

Malware or *malicious code* is any element of software that performs an unwanted function from the perspective of the legitimate user or owner of a computer system. This objective topic focuses on your ability to recognize a specific type of malware from a given scenario, list of symptoms, or general description of an infection or compromise. Malicious code includes a wide range of concepts, including viruses, ransomware, worms, Trojans, rootkits, keyloggers, adware, spyware, bots, RATs (Remote Access Trojan), logic bombs, and backdoors. Following is an overview of each.

Viruses

Viruses are just one example of malicious code, malicious software, or malware. *Viruses* get their name from their biological counterparts. They're programs designed to spread from one system to another through self-replication and to perform any of a wide range of malicious activities. The malicious activities performed by viruses include data deletion, corruption, alteration, and exfiltration. Some viruses replicate and spread so rapidly that they consume most of the available system and network resources, thus performing a type of denial-of-service (DoS) attack (discussed later in this chapter).

Most viruses need a host to latch onto. The host can be a file (as in the case of *common viruses*) or the boot sector of a storage device. Viruses that attach themselves to the boot sector of a storage device (including HDD, SSD, CD/DVD-ROM, Blu-ray, and USB), and thus are loaded in memory when the drive is activated, are known as *boot sector viruses*.

Within these categories, some specific virus types include the following:

Polymorphic viruses *Polymorphic viruses* have the ability to mask their own code using encryption in order to avoid detection by antivirus scanners.

Macro viruses *Macro viruses* live within documents or emails and exploit the scripting capabilities of productivity software.

Stealth viruses *Stealth viruses* attempt to avoid detection by masking or hiding their activities.

Armored viruses *Armored viruses* are any form of malware that has been crafted to avoid detection and make removal difficult. This can involve the use of complex compiling techniques, overly complex coding logic, and abnormal use of memory.

Retroviruses *Retroviruses* are specifically targeted at antivirus systems to render them useless.

Phage viruses *Phage viruses* modify or infect many aspects of a system so they can regenerate themselves from any remaining unremoved parts.

Companion viruses A *companion virus* borrows the root filename of a common executable and then gives itself the `.com` extension in an attempt to get itself launched rather than the intended application.

Multipart or multipartite viruses *Multipart* or *multipartite viruses* perform multiple tasks and may infect a system in numerous ways.

The best technology to serve as a countermeasure against viruses is an antivirus or antimalware scanner that is updated regularly and that monitors all local storage devices, memory, and communication pathways for viral activities. However, it is essential that modifying user behavior to avoid risky activities be a core part of the security strategy. Otherwise, without human risk reduction, no technological protections will be sufficient. Examples of activities to reduce or avoid risk include avoiding downloading software from nonvendor sources, not opening email attachments, and avoiding the use of removable media from other environments.

If a system is infected with a virus, some potential symptoms include corrupted or missing data files, applications that will no longer execute, slow system operation, lag between mouse click and system response, application or system crashes, ongoing hard drive activity, and the system's tendency to be unresponsive to mouse movements or keystrokes. Any of these symptoms could accompany a virus infection; however, they can be symptoms of other malware infections as well.

Crypto-malware

Crypto-malware is any form of malware that uses cryptography as a weapon or a defense. Crypto as a weapon is seen in malware such as ransomware, while crypto as a defense is seen in malware such as polymorphic and armored viruses.

Another potential form of crypto-malware is code that seeks out the encryption keys of encrypted storage devices and then discloses those keys to a remote attacker. The goal or purpose of such malware is to grant the attacker access to otherwise protected content.

Symptoms of crypto-malware infection include the inability to access data, missing data, a system that will not boot, a sluggish system (during the encryption processes), and pop-ups demanding payment to decrypt your data.

Ransomware

Ransomware is a form of malware that takes over a computer system, usually by encrypting user data, in order to hinder its use while demanding payment. Effectively, it's malware that holds a user's data hostage in exchange for a ransom payment. Often, the thieves behind ransomware request payment to be made in untraceable money cards, such as the MoneyPak Green Dot card, or in Bitcoins (a form of digital currency intended to be untraceable).

Countermeasures against ransomware include avoiding risky behaviors, running anti-malware software, and maintaining a reliable backup of your data. Unless absolutely no other option is available to you to regain access to your data, avoid paying the ransom. Paying a ransom to attackers only encourages them to continue their criminal activities.

Symptoms of ransomware infection include the inability to access data, missing data, a system that will not boot, a sluggish system (during the encryption processes), and pop-ups demanding payment to decrypt your data.

Worm

Another form of malware that is closely related to a virus is a worm. *Worms* are self-contained applications that don't require a host file or hard drive to infect. Worms typically are focused on replication and distribution, rather than on direct damage and destruction. Worms are designed to exploit a specific vulnerability in a system (operating system, protocol, service, or application) and then use that flaw to spread themselves to other systems with the same flaw. They may be used to deposit viruses, logic bombs, ransomware, backdoors, or zombies/agents/bots for botnets, or they may perform direct virus-like maelstrom activities on their own.

Countermeasures for worms are the same as for viruses, with the addition of keeping systems patched.

A worm infection may display symptoms that include a slow-to-respond system, applications that no longer will execute, a lack of free space on storage devices, CPU and memory utilization maxed out at 100 percent, system crashes, and abnormal network activity.

Trojan

A *Trojan horse* is a form of malicious software that is disguised as something useful or legitimate. The most common forms of Trojan horses are games and screensavers, but any software can be made into a Trojan. The goal of a Trojan horse is to trick a user into installing it on their computer. This allows the malicious code portion of the Trojan to gain access to the otherwise secured environment. A Trojan is crafted by combining a seemingly benign host file with a malicious payload. It is an integration of technology abuse with social engineering. The victim is tricked into accepting the Trojan on their system because they believe that the only thing they are obtaining is the obvious benign host. However, when the host is used, the malicious payload is released to infect the system. Some of the most common Trojans are tools that install distributed denial-of-service (DDoS), botnet agents, or remote-control backdoors onto systems.

Countermeasures for Trojan horses are the same as for viruses.

Scenarios involving a system becoming infected through Trojan horse delivery of malware can elicit any of the symptoms mentioned for other malware infections (see earlier and later malware concepts), since a Trojan horse can be used to deliver any sort of malicious code. In addition, a Trojan horse may cause system slowdown or unresponsiveness immediately after triggering or launching the Trojan horse while it is delivering the malicious payload.

Rootkit

A *rootkit* is a special type of hacker tool that embeds itself deep within an operating system (OS). The rootkit positions itself at the heart of an OS, where it can manipulate information seen by the OS. Often, a rootkit replaces the OS kernel, shims itself under the kernel, replaces device drivers, or infiltrates application libraries so that whatever information it feeds or hides from the OS, the OS thinks is normal and acceptable. This allows a rootkit to hide itself from detection, prevent its files from being viewed by file management tools, and prevent its active processes from being viewed by task management or process management tools. Thus, a rootkit is a type of invisibility shield. A rootkit can be used to hide other malicious tools and/or perform other functions. A rootkit or other tools hidden by a rootkit can capture keystrokes, steal credentials, watch URLs, take screen captures, record sounds via the microphone, track application use, or grant a remote hacker backdoor access or remote control over the compromised target system.

After a rootkit has infected a system, that system can no longer be trusted or considered secure. There are rootkits that are still undetectable and/or can't be effectively removed. Thus, any rootkit-compromised system can never be fully trusted again. To use a silly analogy: if you're fighting an invisible army, how can you be sure that you've defeated all of the soldiers?

There are several rootkit-detection tools, some of which are able to remove certain rootkits. However, once you suspect a rootkit is on a system, the only truly secure response is to reconstitute or replace the entire computer. *Reconstitution* involves performing a low-level formatting operation on all storage devices on that system, reinstalling the OS and all applications from trusted original sources, and then restoring files from trusted rootkit-free backups. Obviously, the best protection against rootkits is defense rather than response.

There are often no noticeable symptoms or indicators of compromise related to a rootkit infection. Rootkit authors often strive to minimize any noticeable activity that might indicate that a system has been compromised. In the moments after initial rootkit installation there might be some system sluggishness and unresponsiveness as the rootkit installs itself, but otherwise it will actively mask any symptoms.

Keylogger

A *keylogger* is a form of malware that records the keystrokes typed into a system's keyboard. Software keyloggers are often able to record input from both physical keyboards and on-screen keyboards. The captured keystrokes are then uploaded to the attacker for analysis and exploitation.

Many antimalware scanners include signatures for keyloggers; however, a potentially unwanted program (PUP) scanner, such as Malwarebytes, might also be necessary to detect this type of abusive software.

Hardware keyloggers are physical devices attached to the keyboard cable where it connects to the main system. Such devices are not detectable by software and thus require physical inspection to uncover. Some hardware keyloggers can upload captured content via Wi-Fi, Bluetooth, or cellular service, whereas others must be physically retrieved.

A keylogger infection might exhibit sluggish keyboard response, require typing keys twice to get them to be recognized by the system, and cause overall system performance degradation.

Adware

Adware is a variation on the idea of spyware (discussed later in this section). Adware displays pop-up advertisements to users based on their activities, URLs they have visited, applications they have accessed, and so on. Adware is used to customize advertisements to prospective customers. Unfortunately, most adware products arrive on client systems without the knowledge or consent of the user. Thus, legitimate commercial products are often seen as intrusive and abusive adware.

Some forms of adware display offerings for fake or false security products. They often display an animation that seems like the system is being scanned; they may even search for malicious code or intrusion events. The adware then displays a warning that problems were found and the solution is to download a "free" utility to remove or resolve the offense. This type of malware is also known as scareware.

Countermeasures for adware are the same as for spyware and viruses—antimalware software with added specific spyware/adware-scanning tools.

Indicators of adware compromise can include the pop-up display of advertisements even when a web browser is not already running, sluggish system response, and poor mouse responsiveness (especially when clicking on links).

Spyware

Spyware is any form of malicious code or even business or commercial code that collects information about users without their direct knowledge or permission. Spyware can be fully malicious when it seeks to gain information to perform identity theft or credential hijacking. However, many advertising companies use less malicious forms of spyware to gather demographics about potential customers. In either case, the user is often unaware that the spyware tool is present or that it's gathering information that is periodically transmitted to some outside entity. Spyware can collect keystrokes, names of launched applications, local files, sent or received emails and instant messages (IMs), and URLs visited; it can also record audio by turning on the microphone, or even record video by turning on a webcam. Spyware can be deposited by viruses, worms, or Trojan horses, or it can be installed as an extra element from commercial, freeware, or shareware applications.

Countermeasures for spyware are the same as for viruses, with the addition of specific spyware-scanning tools.

Spyware infections may cause noticeable symptoms such as slow system performance, poor keyboard and mouse responsiveness, the appearance of unknown files, and quickly dwindling available storage space.

Bots

The term *botnet* is a shortened form of the phrase *robot network*. It is used to describe a massive deployment of malicious code onto numerous compromised systems that are all remotely controlled by a hacker. A botnet is the culmination of traditional DoS attacks into a concept known as a *distributed denial-of-service (DDoS) attack*. A DDoS attack occurs when a hacker has deposited remote-controlled agents, zombies, or bots onto numerous secondary victims and then uses the deployed bots as a single entity to attack a primary target. (This is covered in more detail later in this chapter, when we review specific attack types.)

Botnets are either directly or indirectly controlled by a hacker. Sometimes the hacker is called a *bot herder*, a *master*, or even a *handler*. Direct control of a botnet occurs when the bot herder sends commands to each bot. Therefore, bots have a listening service on an open port waiting for the communication from the bot herder. Indirect control of a botnet can occur through any intermediary communication system, including Internet Relay Chat (IRC), IM, File Transfer Protocol (FTP), email, the Web, blogging, Facebook, Twitter, and so on. When indirect control is used, the bots access an intermediary communication service for messages from the bot herder. The intermediary communication service is often named a "command and control center," but instead of being a complex controlling interface, it is simply the locus of connection between the attacker and the bots where information is exchanged.

Botnets are possible because most computers around the world are accessible over the Internet, and many of those computers have weak security. A botnet creator writes their botnet code to exploit a common vulnerability in order to spread the botnet agent far and wide—often using the same techniques used by viruses, worms, and Trojan horses. Botnets typically include thousands (if not hundreds of thousands) of compromised secondary victims. The secondary victims are the hosts of the botnet agent itself and aren't affected or damaged beyond the initial intrusion and planting of the botnet agent. The hackers want the secondary victims fully functional so that when they launch their botnet attack against the primary victim, they can use all the resources of the secondary victims against the primary target.

A botnet can be used to perform any type of malicious activity. Although they're most often used to perform DoS flooding attacks, botnets can also be used to transmit spam, perform massively distributed parallel processing to crack passwords or encryption keys, perform phishing attacks, capture network packets, or perform any other conceivable activity.

The best defense against a botnet is to keep your systems patched and hardened and to not become the host of a botnet agent (in other words, don't become a secondary victim). Strict outbound firewall rules, spoofed source address filtering, and web content filtering on a unified threat management (UTM) device are also effective countermeasures. In addition, most antivirus software and antispyware/adware tools include well-known botnet agents in their detection databases.

If you're the primary victim of a botnet flooding attack, there is little you can do to stop the attack. Your responses are often limited to disconnecting from the Internet, contacting your ISP, and reporting the incident to law enforcement. There are several DDoS filtering services, which range from free services to quite expensive enterprise-class services.

The indicators of botnet compromise can include slow system performance, high levels of CPU and memory utilization, high levels of abnormal network traffic, strange files appearing on storage devices, unknown processes running, and odd program windows appearing on the desktop.

RAT

A remote-access Trojan (*RAT*) is a form of malicious code that grants an attacker some level of remote-control access to a compromised system. Often the remote-control backdoor component is hidden inside a host file that is linked to some current popular concept, such as a new movie, music album, or game. Once the victim uses or opens the host, the remote-control malware is installed on their system and a notification is sent to the attacker. Most RATs then initiate an outbound connection to the attacker's waiting system to grant them access to manipulate the victim's data and system operations.

RAT infections may result in noticeable symptoms such as odd network communications and traffic levels; a system that will not auto-engage the screensaver or timed sleep mode; higher levels of drive, CPU, and memory activity; and the appearance of unknown files on storage devices.

Logic bomb

A *logic bomb* is a form of malicious code that remains dormant until a triggering event occurs. The triggering event can be a specific time and date, the launching of a specific program, typing in a certain keystroke combination, or the accessing of a specific URL (such as your online banking logon page). Logic bombs can perform any malicious function the programmer wishes, from causing system crashes, to deleting data, to altering configurations, to stealing authentication credentials.

A logic bomb can also be a fork bomb, which triggers a duplication event where the original code is cloned and launched. Then, each of the new clones forks itself again. This forking/cloning process repeats until the system crashes due to complete resource consumption by the malware. A fork bomb also works by consuming storage space or using up the network bandwidth.

Symptoms of logic bomb compromise could include an abrupt change in system performance, crashing of applications or the system, and a loss of storage device free space.

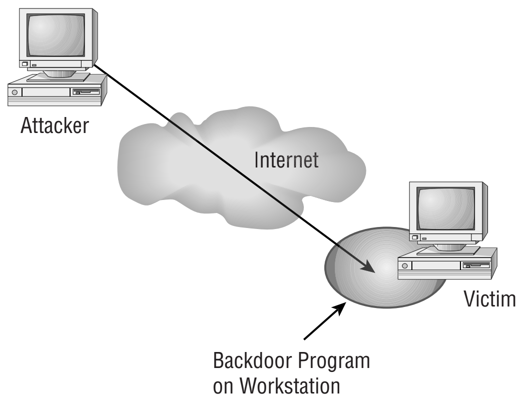
Backdoor

The term *backdoor* can refer to two types of problems or attacks on a system. The first and oldest type of backdoor was a developer-installed access method that bypassed all security restrictions. The backdoor was a special hard-coded user account, password, or command sequence that allowed anyone with knowledge of the access hook (sometimes called a *maintenance hook*) to enter the environment and make changes. This sounds great from a developer's perspective, especially during the coding and debugging process. Unfortunately, such programming shortcuts are often forgotten about when the product nears completion; thus, they end up in the final product. Fortunately, once a backdoor is discovered in a released product, the vendor usually releases a patch to remove the backdoor code from the installed product. The possible presence of backdoors is another good reason to stay current with vendor-released updates and patches.

The second meaning of backdoor is a hacker-installed remote-access client. These small, maliciously purposed tools can easily be deposited on a computer through a Trojan horse, a virus, a worm, a website mobile code download, or even as part of an intrusion activity. Once active on a system, the tool opens access ports and waits for an inbound connection. Thus, a backdoor serves as an access portal for hackers so that they can bypass any security restrictions and gain (or regain) access to a system. Some common backdoor tools include Back Orifice, NetBus, and Sub7 (all of which function on Windows). These and other common backdoor tools are detected and removed by virus scanners and spyware scanning tools.

Figure 1.1 shows a backdoor attack in progress.

FIGURE 1.1 A backdoor attack in progress



Preemptive measures against backdoors include restricting mobile code from being automatically downloaded to your systems, using software policies to prevent unauthorized software from being installed, monitoring inbound and outbound traffic, and requiring software and driver signing.

A backdoor compromise may elicit noticeable symptoms such as an unresponsive system, applications opening or closing seemingly on their own, abnormal network connections and activity, and missing or new files.

Exam Essentials

Understand viruses. Viruses are programs that are designed to spread from one system to another through self-replication and to perform any of a wide range of malicious activities.

Understand crypto-malware. Crypto-malware is any form of malware that uses cryptography as a weapon or a defense.

Understand ransomware. Ransomware is a form of malware that aims to take over a computer system in order to block its use while demanding payment.

Understand worms. Worms are designed to exploit a single flaw in a system (operating system, protocol, service, or application) and then use that flaw to replicate themselves to other systems with the same flaw.

Understand Trojan horses. A Trojan horse is a form of malicious software that is disguised as something useful or legitimate.

Understand rootkits. A rootkit is a type of malicious code that fools the OS into thinking that active processes and files don't exist. Rootkits render a compromised system completely untrustworthy.

Understand keyloggers. A keylogger is a form of malware that records the keystrokes typed into a system's keyboard.

Understand spyware and adware. Spyware gathers information about users and may employ that information to customize advertisements or steal identities. Adware gathers information about users and uses it to direct advertisements to the user. Both spyware and adware are usually unwanted software that gathers information without authorization.

Understand botnets. A botnet is a network of robots or malicious software agents controlled by a hacker in order to launch massive attacks against targets.

Understand a RAT. A remote-access Trojan (RAT) is a form of malicious code that grants an attacker some level of remote-control access to a compromised system.

Understand logic bombs. A logic bomb is a form of malicious code that remains dormant until a triggering event occurs. The triggering event can be a specific time and date, the launching of a specific program, or the accessing of a specific URL.

Understand backdoor attacks. There are two types of backdoor attacks: a developer-installed access method that bypasses any and all security restrictions, or a hacker-installed remote-access client.

Understand malicious code countermeasures. The best countermeasure to viruses and other malicious code is an antivirus scanner that is updated regularly and that monitors all local storage devices, memory, and communication pathways for malicious activity. Other countermeasures include avoiding downloading software from the Internet, not opening email attachments, and avoiding the use of removable media from other environments.

1.2 Compare and contrast types of attacks.

Any computer system connected to any type of network is subject to various types of attacks. The rate at which networked systems are attacked is increasing at an alarming rate. Even systems that aren't connected to the Internet, such as those isolated in a private network, may come under attack. There are myriad ways to attack a computer system. Your familiarity with a modest collection of these attacks and how to respond to them is an essential skill for the Security+ exam. The following sections discuss common attack methods.

Social engineering

Social engineering is a form of attack that exploits human nature and human behavior. Social engineering attacks take two primary forms: convincing someone to perform an unauthorized operation or convincing someone to reveal confidential information. For example, the victim may be fooled into believing that a received email is authoritative (such as an email hoax), that a person on the phone is someone to be respected and obeyed (such as someone claiming to be from tech support or a manager offsite), or that a person with them is who they claim to be (such as an air-conditioning [AC] repair technician). In just about every case, in social engineering the attacker tries to convince the victim to perform some activity or reveal a piece of information that they shouldn't. The result of a successful attack is information leakage or the attacker being granted logical or physical access to a secure environment.

Any form of advertisement could be considered a form of social engineering attack—ads appeal to you in an attempt to get you to purchase or use a product or service. Although an advertisement's motivation is profit, the motives for most social engineering attacks are more malevolent. In fact, hackers now have access to sophisticated technology to assist in their social engineering endeavors.

One such tool is the Social Engineering Toolkit (SET). As you can see on the `http://social-engineer.org` website, SET was specifically designed to perform advanced attacks against the human element. It integrates with the Metasploit framework to allow an attacker to take control of a remote computer by enticing the soon-to-be victim to click a pop-up of some sort. For instance, a gamer playing the latest version of the newest hot online video game could receive a pop-up stating that there is temporary Internet congestion. It might then say, "Please select Stay Online if performance is acceptable or select Disconnect to disconnect and reconnect." Either selection results in the attacker's code being run and possibly in the exploitation of the system. The user-interaction portion of the attack is why this is referred to as the Social Engineering Toolkit.

Here are some example scenarios of common social engineering attacks:

- A worker receives an email warning about a dangerous new virus spreading across the Internet. The message directs the worker to look for a specific file on the hard drive and delete it, because it indicates the presence of the virus. Often, however, the identified file is really an essential file needed by the system.
- A website claims to offer free temporary access to its products and services, but it requires web browser and/or firewall alterations in order to download the access software.
- A secretary receives a phone call from a person claiming to be a client who is running late to meet the CEO. The caller asks for the CEO's private cell phone number in order to call them.
- The helpdesk receives a call from an outside line. The caller claims to be a manager of a department who is currently involved in a sales meeting in another city. The caller claims to have forgotten their password and needs it to be reset so that they can log in remotely to download an essential presentation.
- Someone who looks like an AC repair technician enters the office and claims a service call was received for a malfunctioning unit in the building. The "technician" is sure the unit can be accessed from inside your office work area and asks to be given free rein to repair the AC system.
- An unexpected pop-up requires a selection of some sort.

These are just a few examples of possible social engineering attacks. They may also be legitimate and benign occurrences, but you can see how they could mask the motives and purposes of an attacker.

Methods to protect against social engineering include the following:

- Training personnel about social engineering attacks and how to recognize common signs
- Requiring authentication when performing activities for personnel over the phone
- Defining restricted information that is never communicated over the phone
- Always verifying the credentials of a repair person and verifying that a real service call was placed by authorized personnel
- Never following the instructions of an email without verifying the information with at least two independent and trusted sources
- Always erring on the side of caution when dealing with anyone you don't know or recognize, whether in person, over the phone, or over the Internet/network

The only real defense against social engineering attacks is user education and awareness training. A healthy dose of paranoia and suspicion will help users detect or notice social engineering attack attempts. Training should include role playing and numerous examples of the various forms of social engineering attacks.

Phishing

Phishing is a form of social engineering attack focused on stealing credentials or identity information from any potential target. It is based on the concept of fishing for information. Phishing is employed by attackers to obtain sensitive information such as usernames, passwords, credit card details, or other personally identifiable information by masquerading as a trustworthy entity (a bank, a service provider, or a merchant, for example) in electronic communication (usually email). Phishing can be waged in numerous ways using a variety of communication media, including email, the Web, live discussion forums, IM, message boards, and so on.

To defend against phishing attacks, end users should be trained to avoid clicking any link received via email, IM, or social network message. Instead, the user should visit the supposed site by using a preestablished bookmark or by searching for the site by name. If, after accessing their account on the site, a duplicate message does not appear in the online messaging or alert system, the original message is likely an attack or a fake. Any such false communications should be reported to the targeted organization, and then the message should be deleted.

All forms of phishing take advantage of people's willingness to extend trust to apparently legitimate third parties without applying rules of basic, commonsense information security (the most germane of these principles here are "never open unexpected email attachments" and "never share sensitive information via email").

Spear phishing

Spear phishing is a more targeted form of phishing where the message is crafted and directed specifically to a group of individuals, rather than being just a blind broadcast to anyone. Often, attackers will first compromise an online or digital business in order to steal their customer database. Then, false messages are crafted to seem like a communication from the compromised business, but with falsified source addresses and incorrect URLs. The hope of the attack is that someone who already has an online/digital relationship with an organization is more likely to fall for the false communication. If the victim responds, then the followup messages or the website they access is crafted to elicit their personal information in order to perform account takeover or full-fledged identity theft.

Whaling

Whaling is a form of phishing that targets specific high-value individuals (by title, by industry, from media coverage, and so forth), such as C-level executives or high-net-worth clients, and sends messages tailored to the needs and interests of those individuals. Whaling attacks require significantly more research, planning, and development on the part of the attackers in order to fool the victim. But a successful attack can be a significant payoff for the malicious hacker.

Vishing

Vishing is phishing done via *Voice-over-IP* (*VoIP*) services. VoIP is a technology that allows phone call–like conversations to take place over TCP/IP networks. Many companies and individuals use VoIP phones instead of traditional landline phones. The victims of vishing do not have to be using VoIP. Instead, the attack originates from a VoIP service. This allows the attacker to be located anywhere in the world and make a free phone call to the victim.

Vishing is simply another form of phishing attack. The main problem with vishing is that tracing the source or origin of the attacks is much more complicated, if not impossible. Thus, it's more important than ever to be suspicious of phone calls, even those with correct caller ID. Everyone should take the extra effort to verify the caller, or hang up on them and then call the claimed entity back using a known trusted phone number, such as the one on the back of your credit card or from the entity's official website. Users should be trained to be careful about volunteering information when prompted by a caller, such as being asked to provide account numbers, account passwords, secret PINs, billing address, and so on. These are fine to disclose to the valid entity when a user originates the call, but when someone else calls, there is no way to fully verify that they are the claimed entity.

Tailgating

Tailgating occurs when an unauthorized entity gains access to a facility under the authorization of a valid worker but without their knowledge. This attack can occur when a worker uses their valid credentials to unlock and open a door, then walks on into the building as the door closes, granting the attacker the opportunity to stop the door from closing and sneak in without the victim realizing. Tailgating is an attack that does not depend on the consent of the victim, just their obliviousness to what occurs behind them as they walk into a building.

Tailgate prevention by users is very simple. Each and every time a user unlocks or opens a door, they should ensure that it is closed and locked before walking away. This action alone eliminates tailgating. There is social pressure to hold open a door for someone who is walking up behind you, but this courtesy should not be extended to include secure entry points.

A problem similar to tailgating is *piggybacking*. Piggybacking occurs when an unauthorized entity gains access to a facility under the authorization of a valid worker but with their knowledge and consent. This could happen when the intruder feigns the need for assistance by holding a large box or lots of paperwork and asks someone to "hold the door." The goal is to distract the victim while the attacker gains access in order to prevent the victim from realizing that the attacker did not provide their own credentials.

Users should be trained to watch out for this type of attack. When someone asks for assistance in holding open a secured door, users should ask for proof of authorization or offer to swipe the person's access card on their behalf. This reduces the chance of an outsider bluffing their way into your secured areas.

In addition to user behavior changes, mantraps, turnstiles, and security guards all reduce tailgating and piggybacking significantly.

Impersonation

Impersonation is the act of taking on the identity of someone else. This can take place in person, over the phone, or through any other means of communication. The purpose of impersonation is to fool someone into believing you have the claimed identity so you can use the power or authority of that identity. Impersonation is a common element of social engineering. Impersonation can also be known as *masquerading*.

A form of impersonation known as *pretexting* can occur when an individual describes a false situation as a pretext for the social engineering attack.

Dumpster diving

Dumpster diving is the act of digging through trash, discarded equipment, or abandoned locations in order to obtain information about a target organization or individual. Although discovering confidential documentation or secret information would be a welcomed bonus to attackers, they are looking for more mundane documentation. Typical collected items include old calendars, calling lists, meeting notes, discarded forms, product boxes, user manuals, sticky notes, printed reports, or the test sheet from a printer. Dumpster diving can provide an attacker with information that could make social engineering attacks easier or more effective.

To prevent dumpster diving, or at least reduce its value, all documents should be shredded and/or incinerated before being discarded. Additionally, no storage media should ever be discarded in the trash; use a secure disposal technique or service.

Shoulder surfing

Shoulder surfing occurs when someone is able to watch a user's keyboard or view their display. This could allow them to learn a password or see information that is confidential, private, or simply not for their eyes. Often, shoulder surfing is stopped by dividing worker groups by sensitivity levels using locked doors. Additionally, users should not orient their displays to be visible through windows (from outside) or walkways/doorways (for internal issues). And they should not work on sensitive data while in a public space, such as a coffee shop or on a plane.

Hoax

A *hoax* is a form of social engineering designed to convince targets to perform an action that will cause problems or reduce their IT security. A hoax is often an email that proclaims some imminent threat is spreading across the Internet and that you must perform certain tasks in order to protect yourself. Victims may be instructed to delete files or change configuration settings, which results in a compromised OS, a nonbooting OS, or a reduction in their security defenses. Additionally, hoax emails often encourage the victim to forward the message to all their contacts in order to "spread the word."

Watering hole attack

A *watering hole attack* is a form of targeted attack against a region, a group, or an organization. The attack is performed in three main phases. The first phase is to observe the target's habits. The goal is to discover a common resource, site, or location that one or more members of the target frequent. This location is considered the watering hole. The second phase is to plant malware on watering hole systems. The third phase is to wait for members of the target to revisit the poisoned watering hole and then bring the infection back into the group. The name is derived from the concept of wiping out an animal population by poisoning its primary water source. This technique is fairly effective at infiltrating groups that are well secured, are difficult to breach, or operate anonymously. For an example of a watering hole attack performed by the FBI, see `www.wired.com/threatlevel/2013/09/freedom-hosting-fbi/`.

Principles (reasons for effectiveness)

Social engineering works so well because we're human. The principles of social engineering attacks are designed to focus on various aspects of human nature and take advantage of them. Although not every target succumbs to every attack, most of us are vulnerable to one or more of the following common social engineering principles.

Authority

Authority is an effective technique because most people are likely to respond to authority with obedience. The trick is to convince the target that the attacker is someone with valid authority. That authority can be from within an organization's internal hierarchy or from an external recognized authority, such as law enforcement, technical support, pest extermination, utility inspection, debt collection, and so on. Some attackers claim their authority verbally, and others assume authority by wearing a costume or uniform.

Intimidation

Intimidation can sometimes be seen as a derivative of the authority principle. Intimidation uses authority, confidence, or even the threat of harm to motivate someone to follow orders or instructions. Often, intimidation is focused on exploiting uncertainty in a situation where a clear directive of operation or response isn't defined. The attacker attempts to use perceived or real force to bend the will of the victim before the victim has time to consider and respond with a denial.

Consensus

Consensus or *social proof* is the act of taking advantage of a person's natural tendency to mimic what others are doing or are perceived as having done in the past. For example, bartenders often seed their tip jar with money to make it seem as if previous patrons were appreciative of the service. People visiting a tourist spot might carve their name in a railing because many previous visitors' names are present. People will stop walking down the street and join a crowd, just to see what is going on. As a social engineering principle, the attacker attempts to convince the victim that a particular action or response is preferred in order to be consistent with social norms or previous occurrences. For example, an attacker may claim that a worker who is currently out of the office promised a large discount on a purchase and that the transaction must occur now with you as the salesperson.

Scarcity

Scarcity is a technique used to convince someone that an object has a higher value based on the object's scarcity. For example, shoppers often feel motivated to make a purchase because of a limited-time offer, due to a dwindling stock level, or because an item is no longer manufactured.

Familiarity/liking

Familiarity or *liking* as a social-engineering principle attempts to exploit a person's native trust in that which is familiar. The attacker often tries to appear to have a common contact or relationship with the target, such as mutual friends or experiences, or uses a facade to take on the identity of another company or person. If the target believes a message is from a known entity, such as a friend or their bank, they're much more likely to trust in the content and even act or respond.

Trust

Trust as a social engineering principle involves an attacker working to develop a relationship with a victim. This may take seconds or months, but eventually the attacker attempts to use the value of the relationship (the victim's trust in the attacker) to convince the victim to reveal information or perform an action that violates company security.

Urgency

Urgency often dovetails with scarcity, because the need to act quickly increases as scarcity indicates a greater risk of missing out. Urgency is often used as a method to get a quick response from a target before they have time to carefully consider or refuse compliance.

Application/service attacks

Social engineering is not the only form of attack faced by modern environments. A wide variety of attacks and exploitations are used by attackers to exfiltrate data or gain logical or physical access to our organizations. In this section, I discuss several examples of attacks that take advantage of information technology. Keep in mind a phrase attributed to the NSA: "Attacks always get better; they never get worse," meaning that while you may not be vulnerable to a particular attack today, you might be tomorrow. It is also the case that new attacks are being developed by attackers, so you may not even be aware of a new method of exploitation until after your systems have fallen victim to it.

Arbitrary Code Execution/Remote Code Execution

Arbitrary code execution is the ability to run any software—particularly malicious shell code—on a target system. This ability is usually the focus of most hacker exploits and attacks, such as those mentioned in this section. When combined with privilege escalation, a hacker's capacity to arbitrarily run any software of their choosing at an administrator, root, or system level means they have the open-ended ability to perform any task on the system. Often, this capability is established using a remote attack as opposed to a local attack (an attack run on an authorized system from within an authorized user account). A remote exploitation of arbitrary code execution is also called *remote code execution*.

DoS

Denial of service (*DoS*) is a form of attack that has the primary goal of preventing the victimized system from performing legitimate activity or responding to legitimate traffic. There are two basic types of DoS attack. The first form exploits a weakness, an error, or a standard feature of software to cause a system to hang, freeze, consume all system resources, and so on. The end result is that the victimized computer is unable to process any legitimate tasks. The second form floods the victim's communication pipeline with garbage network traffic. Such garbage traffic can be false responses to nonexistent requests, partial establishment of a TCP session, or repeated requests for data from a service or application. The end result is that the victimized computer is unable to send or receive legitimate network communications. In any case, the victim is denied the ability to perform normal operations (services).

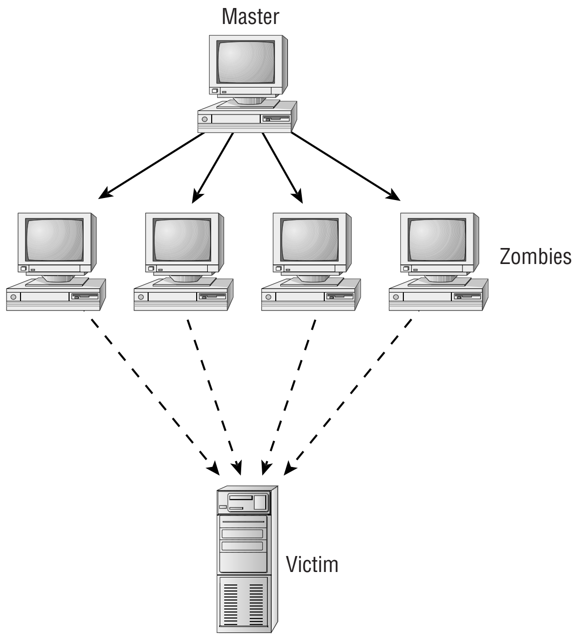
DoS isn't a single attack but rather an entire class of attacks. Some attacks exploit flaws in OS software, whereas others focus on installed applications, services, or protocols. Some attacks exploit specific protocols, including Internet Protocol (IP), Transmission Control Protocol (TCP), Internet Control Message Protocol (ICMP), and User Datagram Protocol (UDP).

DoS attacks typically occur between one attacker and one victim. However, they don't have to be waged in that simple a manner. Most DoS attacks employ some form of intermediary system (usually an unwilling and unknowing participant) in order to hide the attacker from the victim. For example, if an attacker sends attack packets directly to a victim, it's possible for the victim to discover who the attacker is. This is made more difficult, although not impossible, through the use of *spoofing* (discussed later in this chapter).

The next generation of DoS attacks is known as *distributed denial-of-service (DDoS)* attacks. These types of DoS attacks are waged by first compromising or infiltrating one or more intermediary systems that serve as launch points or attack platforms. These intermediary systems are commonly referred to as *secondary victims*. The attacker installs remote-control tools, often called *bots*, *zombies*, or *agents*, onto these systems. Then, at an appointed time or in response to a launch command from the attacker, the DoS attack is conducted against the victim, as shown in Figure 1.2. In this manner, the victim may be able to discover the zombied system(s) that are causing the DoS attack but probably won't be able to track down the actual attacker. Recently, such deployments of many bots or zombies across numerous unsuspecting secondary victims have become known as *botnets* (see the earlier section "Botnets").

In addition to DoS and DDoS, there is a third form known as *distributed reflective denial-of-service (DRDoS)*. This form of attack employs an *amplification* or *bounce* network that is an unknowing participant, unfortunately able to receive broadcast messages and create message responses, echoes, or bounces. In effect, the attacker sends spoofed message packets to the amplification network's broadcast address. This causes each single inbound received packet to be distributed to all the hosts in that network (which could be in the 10,000 or 100,000 range). Each host then responds to each packet, but because the source of the original packet was falsified, the response goes to the victim instead of the true sender (the attacker). So, what originated from the attacker as a single packet is transformed into numerous packets exiting the amplification network and ultimately flooding the victim's communication link.

FIGURE 1.2 DDoS attack



There are numerous specific DoS, DDoS, and DRDoS attack tools and methods. Here are a few that you should be able to recognize:

Smurf This form of DRDoS uses ICMP echo reply packets (ping packets). The attacker sends ICMP Type 8 echo request packets to several intermediary networks' broadcast addresses with the source IP address set to the primary victim. This causes multiple ICMP Type 0 replies to be sent to the victim. A smurf attack is also known as an amplification attack. See Figure 1.3 for an example.

Fraggle This form of DRDoS uses UDP packets commonly directed to port 7 (echo port) or 19 (chargen [character generator] port).

SYN flood This type of attack is an exploitation of a TCP three-way handshake. Every TCP session starts with the client sending a SYN (synchronize) packet to a server, the server responding with a SYN/ACK (synchronize/acknowledgment) packet, and the client sending a final ACK packet. The attack consists of the attacker posing as a client and sending numerous SYN packets but never any final ACK packets. This causes the server to consume all network resources by opening numerous incomplete communication sessions. Figure 1.4 shows an example of a TCP SYN flood attack.

FIGURE 1.3 A smurf attack underway against a network

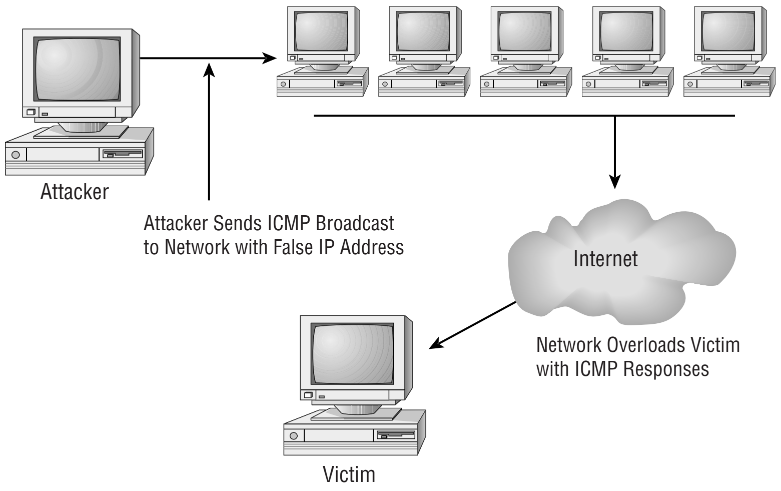
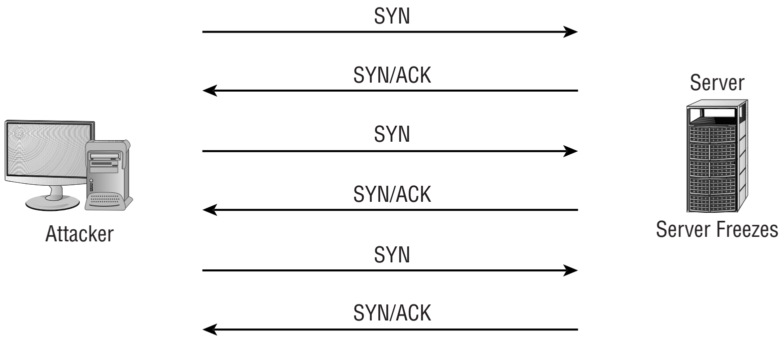

Attacker
Attacker Sends ICMP Broadcast to Network with False IP Address
Internet
Network Overloads Victim with ICMP Responses
Victim

FIGURE 1.4 TCP SYN flood attack



Ping of death The attacker sends oversized ping packets to the victim. The victim doesn't know how to handle invalid packets, and it freezes or crashes.

Xmas attack The *Xmas attack* is actually an Xmas scan. It's a form of port scanning that can be performed by a wide number of common port scanners, including Nmap, Xprobe, and hping2. The Xmas scan sends a TCP packet to a target port with the flags of URG, PSH, and FIN all turned on. This creates a flag byte of 00101001 in the TCP header, which is said to be representative of alternating flashing lights on a Christmas tree. According to the TCP specifications, ports should ignore any invalid construction of a packet if the port is open and send an RST back if the port is closed. This is true of all systems except for Windows OSs, which send RSTs for many invalid packets even if the port is open. An Xmas attack (or scan) occurs when someone sends Xmas-flagged packets to one or more ports on a computer. If the level of scanning packets is significant, this can affect the performance of the targeted system or consume some or all of the available bandwidth. Thus, an Xmas scan can escalate to a DoS and thus be considered an Xmas attack.

SYN floods, teardrops, land attacks, ping floods, pings of death, bonks, and boinks are typically labeled DoS attacks, but they can be waged as a DDoS if the attacker compromises several intermediary systems and uses those as launching points to attack the victim. Fortunately, most of the basic DoS attacks that exploit error-handling procedures (such as ping of death, land attack, teardrop, bonk, boink, and so on) are now automatically handled by improved versions of the protocols installed in the OS. However, many of the current DDoS and DRDoS attacks aren't as easy to safeguard against.

Some countermeasures and safeguards against these attacks are as follows:

- Work out a response plan with your ISP.
- Add firewalls, routers, and intrusion detection systems (IDSs) that detect DoS traffic and automatically block the port or filter out packets based on the source or destination address.
- Disable echo replies on external systems.
- Disable broadcast features on border systems.
- Block spoofed packets from entering or leaving your network.
- Keep all systems patched with the most current security updates from vendors.

Unfortunately, as security professionals develop better defenses, preventions, and detections of the various types of DoS attacks, so hackers are actively developing new means and methods of waging attacks that get around those defenses. In the fall of 2016, the most significant DDoS flooding attack ever took place in response to a blog posting by Brian Krebs on his site `https://www.krebsonsecurity.com/`, where he revealed the "secret" of the existence of DoS as a Service. This attack generated a peak load of 620 Mbps. For details about this attack and related concerns, check out `https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/` and `https://krebsonsecurity.com/2017/02/how-google-took-on-mirai-krebsonsecurity/`.

DDoS

Distributed denial of service (DDoS) was discussed in the previous section.

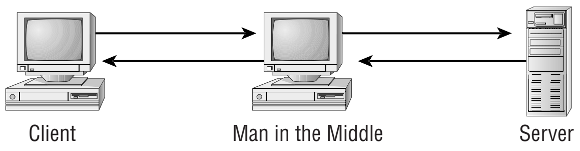
Man-in-the-middle

A *man-in-the-middle attack* is a communications eavesdropping attack. Attackers position themselves in the communication stream between a client and server (or any two communicating entities). The client and server believe that they're communicating directly with each other—they may even have secured or encrypted communication links. However, the attacker can access and potentially modify the communications.

Man-in-the-middle (MitM) attacks range from very simple to quite complex. Some MitM attacks exploit DHCP weaknesses to distribute false IP configurations, such as defining the attack system's IP address as the victim's default gateway. Other forms of MitM attacks focus on poisoning name-resolution systems—such as Domain Name System (DNS), Address Resolution Protocol (ARP), NetBIOS, and Windows Internet Name Service (WINS). Still other MitM attacks include the use of false proxy server settings or using MAC (media access control) address spoofing. Any form of MitM may fool the client into perceiving the attacker as the server and fool the server into perceiving the attacker as the client, or simply cause the attacker to be a transparent node along the communication pathway. When that charade is successful, the client submits its logon credentials to the fake server (the masked attacker), which in turn sends the credentials to the actual server while masquerading as the actual client. As a result, the client establishes a communication link (maybe even an encrypted link) with the attacker, and the attacker establishes a communication link with the server. As data is transmitted in either direction between the true client and server systems, the attacker can read and access all the data and can choose to modify the traffic to further the subterfuge.

Figure 1.5 shows a man-in-the-middle attack.

FIGURE 1.5 A man-in-the-middle attack occurring between a client and a web server



Such attacks are usually most successful when routing and name-resolution systems are first compromised in order to position the attacker before the client-to-server communication is initiated. However, in some cases man-in-the-middle attacks can be conducted against existing client-server communication links (usually assuming they aren't encrypted). One situation where this is possible is with open wireless connections. A deauthentication packet can be sent by an attacker to a wireless client victim; then, as they attempt to reconnect, the attacker fools the victim system into establishing a connection through it, instead of linking up with the valid base station. Even this style of MitM can be effective because it takes only a fraction of a second for the entire process to occur, and unless the short-lived disconnect interrupts an active data transfer, the user won't even notice the event.

Countermeasures to man-in-the-middle attacks include strong encryption protocols (such as IPsec, SSH, and TLS) and the use of strong authentication, such as Domain Name System Security Extensions (DNSSEC) and mutual certificate authentication.

Related to man-in-the-middle is the transitive access attack, or exploitation. Transitive access is a potential backdoor or way to work around traditional means of access control. The idea is that user A can use process B, and process B can use or invoke process C, and process C can access object D (see Figure 1.6). If process B exits (or is otherwise inaccessible) before process C completes, process C may return access to object D back to user A, even if user A doesn't directly or by intent have access to object D (see Figure 1.7). Some forms of access control don't specifically prevent this problem.

All subject to object accesses should be validated before access is granted, rather than relying on previous verifications.

FIGURE 1.6 Transitive access

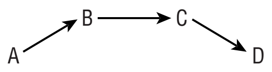
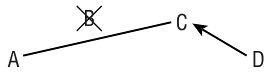


FIGURE 1.7 A transitive access exploit



Buffer overflow

Software exploitation attacks are directed toward known flaws, bugs, errors, and oversights, or toward normal functions of the OS, protocols, services, or installed applications. One of the most common forms of software exploitation is a *buffer overflow attack*.

A buffer overflow attack occurs when an attacker submits data to a process that is larger than the input variable is able to contain. Unless the program is properly coded to handle excess input, the extra data is dropped into the system's execution stack and may execute as a fully privileged operation. Buffer overflow attacks can result in system crashes, corrupted data, user privilege escalation, or just about anything a hacker can think of. The only countermeasures to buffer overflow attacks are to patch the software when issues are discovered and to properly code software to perform input-validation checks before accepting input for processing.

NOTE If you are the user of open source software, then—assuming you know how to code—you have the opportunity to fix flawed code yourself, rather than having to rely on the original programmer or vendor.

Once a weakness is discovered in software, a hacker can craft an exploit or attack tool. These tools are easily accessible and widely distributed on the Internet. They allow anyone to grab the tool and point it at a victim they wish to attack, even when they have neither the knowledge of how the attack actually works nor the skill to craft an attack tool themselves. Those who only use preexisting exploitation tools are known as *script kiddies*.

A buffer overflow occurs when a program receives input that is larger than it was designed to accept or process. The extra data received by the program is shunted over onto the CPU without any security restrictions; it's then allowed to execute (assuming it's a valid command, script, system call, and so on) with system-level privileges. A hacker can achieve many possible results with a buffer overflow: crashing a program, freezing or crashing a system, opening a port, disabling a service, creating a user account, elevating the privileges of an existing user account, accessing a website, or executing a utility. Clever attackers can do just about anything they wish if they can execute a command or script with unrestricted access to a system.

Sometimes a buffer overflow attack can be considered a form of DoS attack, because a buffer overflow occurs when a system receives more data than it can handle (a bit like a flooding attack). This is especially true when the buffer overflow event prevents a system from processing legitimate data or requests.

Poor programming quality controls and a lack of input validation checks in software lead to buffer overflow attacks. Unfortunately, most commercial software is vulnerable to buffer overflow attacks; web server software is attacked most frequently. Fortunately, buffer overflow vulnerabilities are often easily patched with vendor updates or by skilled users when using open source software.

Injection

An *injection attack* is any exploitation that allows an attacker to submit code to a target system in order to modify its operations and/or poison and corrupt its data set. There are a wide range of potential injection attacks. Typically an injection attack is named after the type of backend system it takes advantage of or the type of payload delivered (injected) onto the target. Examples include SQL injection, LDAP injection, XML injection, command injection, HTML injection, code injection, and file injection. A few of these are presented in more detail in this section.

SQL injection attacks are even riskier than XSS attacks (see the following section) from an organization's perspective, because the targets of a SQL injection attack are organizational assets, whereas the targets of an XSS attack are customers or visitors to a website. SQL injection attacks use unexpected input to alter or compromise a web application. However, instead of using this input to attempt to fool a user, SQL injection attacks use it to gain unauthorized access to an underlying database and related assets.

In the early days of the web, all web pages were *static*, or unchanging. Webmasters created web pages containing information and placed them on a web server, where users could retrieve them using their web browsers. The web quickly outgrew this model because users wanted the ability to access customized information based on their individual needs. For example, visitors to a bank website aren't interested only in static pages containing information about the bank's locations, hours, and services. They also want to retrieve dynamic content containing information about their personal accounts. Obviously, the webmaster can't possibly create pages on the web server for each individual user with that user's personal account information. At a large bank, that would require maintaining millions of pages with up-to-the-minute information. That's where dynamic web applications come into play.

Web applications take advantage of a database to create content on demand when the user makes a request. In the banking example, the user logs in to the web application, providing an account number and password. The web application then retrieves current account information from the bank's database and uses it to instantly create a web page containing the user's current account information. If that user returns an hour later, the web server repeats the process, obtaining updated account information from the database.

What does this mean to you as a security professional? Web applications add complexity to the traditional security model. The web server, as a publicly accessible server, belongs in a separate network zone from other servers, commonly referred to as a *demilitarized zone (DMZ)*. The database server, on the other hand, isn't meant for public access, so it belongs on the internal network or at least a secured subnet separated from the DMZ. The web application needs access to the database, so the firewall administrator must create a rule allowing access from the web server to the database server. This rule creates a potential path for Internet users to gain access to the database server.

If the web application functions properly, it allows only authorized requests to the database. However, if there is a flaw in the web application, it may let individuals tamper with the database in an unexpected and unauthorized fashion through the use of SQL injection attacks. These attacks allow a malicious individual to perform SQL transactions directly against the underlying database. SQL injection attacks might enable an attacker to bypass authentication, reveal confidential data from database tables, change existing data, add new records into the database, destroy entire tables or databases, and even gain command line–like access through certain database capabilities (such as command shell stored procedures).

You can use two techniques to protect your web applications against SQL injection attacks:

Perform input validation. Input validation lets you limit the types of data a user provides in a form. There are numerous variations of input injection or manipulation attacks that require a broad-spectrum defense approach, including whitelisting and blacklisting filters. The primary forms of input sanitization that should be adopted include limiting the length of input, filtering on known malicious content patterns, and escaping *metacharacters*.

Metacharacters

Metacharacters are characters that have been assigned special programmatic meaning. Thus, they have special powers that standard, normal characters do not have. There are many common metacharacters, but typical examples include single and double quotation marks; open/close square brackets; the backslash; the semicolon; the ampersand; the caret; the dollar sign; the period, or dot; the vertical bar, or pipe symbol; the question mark; the asterisk; the plus sign; open/close curly braces; and open/close parentheses: ' " [ ] \ ; & ^ $. | ? * + { } ( ).

Escaping a metacharacter is the process of marking the metacharacter as merely a normal or common character, such as a letter or number, thus removing its special programmatic powers. This is often done by adding a backslash in front of the character (\&), but there are many ways to escape metacharacters based on the programming language or execution environment.

Limit account privileges. The database account used by the web server should have the smallest set of privileges possible. If the web application needs only to retrieve data, it should have that ability only.

Ultimately, SQL injection is a vulnerability of the script used to handle the interaction between a front end (typically a web server) and the backend database. If the script was written defensively and included code to escape (invalidate or reject) metacharacters, SQL injection would not be possible.

LDAP injection is a variation of an input injection attack; however, the focus of the attack is on the backend of an LDAP directory service rather than a database server. If a web server front end uses a script to craft LDAP statements based on input from a user, then LDAP injection is potentially a threat. Just as with SQL injection, sanitization of input and defensive coding are essential to eliminate this threat.

XML injection is another variant of SQL injection, where the backend target is an XML application. Again, input sanitization is necessary to eliminate this threat.

Directory Traversal/Command Injection

A *directory traversal* is an attack that enables an attacker to jump out of the web root directory structure and into any other part of the filesystem hosted by the web server's host OS. A common, but historical, version of this attack was against IIS 4.0, hosted by Windows NT 4.0 Server. The attack used a modified URL to directory-traverse out of the web root, into the main OS folders, in order to access the command prompt executable. For example:

```
http://victim.com/scripts/..%c0%af../..%c0%af../..%c0%af../..%c0%af../
..%c 0%af../..%c0%af../winnt/system32/cmd.exe?/c+tftp+-i+get+exploit.exe
```

This URL includes a UNICODE equivalent of the "change to parent directory" command, which is `../` in ASCII, and also notice it uses the metacharacter of percent (%). This URL not only performed directory traversal, but also granted the attacker the ability to perform command injection. The example shows a command injection triggering a TFTP Get operation to download an exploit tool onto the victim web server. Any command that could be executed under the privileges of the IIS service and be crafted within the limitations of a URL could be used. The example performs a single directory listing of the C root. But with minor tweaking, TFTP commands could be used to download hacker tools to the target and subsequently launch those tools to grant greater remote control or true command shell access. This attack can be stopped with metacharacter escaping or filtering.

Cross-site scripting

Cross-site scripting (XSS) is a form of malicious code-injection attack in which an attacker is able to compromise a web server and inject their own malicious code into the content sent to other visitors. Hackers have discovered numerous and ingenious methods for injecting malicious code into websites via CGI scripts, web server software vulnerabilities, SQL injection attacks, frame exploitation, DNS redirects, cookie hijacks, and many other forms of attack. A successful XSS attack can result in identity theft, credential theft, data theft, financial losses, or the planting of remote-control software on visiting clients.

For the administrator of a website, defenses against XSS include maintaining a patched web server, using web application firewalls, operating a host-based intrusion detection system (HIDS), auditing for suspicious activity, and, most importantly, performing server-side input validation for length, malicious content, and metacharacter filtering. As a web user, you can defend against XSS by keeping your system patched, running antivirus software, and avoiding non-mainstream websites. There are add-ons for some web browsers, such as NoScript for Firefox and uBlock Origin for Chrome, that allow only scripts of your choosing to be executed.

Cross-site request forgery

Cross-site request forgery (XSRF) is an attack that is similar in nature to XSS. However, with XSRF, the attack is focused on the visiting user's web browser more than the website being visited. The main purpose of XSRF is to trick the user or the user's browser into performing actions they had not intended or would not have authorized. This could include logging out of a session, uploading a site cookie, changing account information, downloading account details, making a purchase, and so on. One form of XSRF infects a victim's system with malware that stays dormant until a specific website is visited. Then the malware forges requests as the user in order to fool the web server and perform malicious actions against the web server and/or the client.

One example of an exploit that used XSRF is Zeus, which would hide on a victim's system until the user visited their online bank site; then, after it checked their account balance and determined their bank account number, those details would be sent to the controlling attacker, who would initiate an ACH money transfer to another bank. Thus, this is an example of malware that assists in stealing money directly out of the victim's account.

Website administrators can implement prevention measures against XSRF by requiring confirmations or reauthentication whenever a sensitive or risky action is requested by a connected client. This could include requiring the user to reenter their password, sending a code to the user via text message or email that must be provided back to the website, triggering a phone call–based verification, or solving a CAPTCHA (a mechanism to differentiate between humans and software robots). Another potential protection mechanism is to add a randomization string (called a *nonce*) to each URL request and session establishment and check the client HTTP request header referrer for spoofing. End users can form more secure habits, such as running antimalware scanners; using a HIDS; running a firewall; avoiding non-mainstream websites; always logging off from sites instead of closing the browser, closing the tab, or moving on to another URL; keeping browsers patched; and clearing out temporary files and cached cookies regularly.

Privilege escalation

Privilege escalation occurs when a user is able to obtain greater permissions, access, or privileges than they're assigned by an organization. Privilege escalation can occur accidentally or due to administrative oversight, but usually this term refers to the specific and intentional abuse of a system to steal access.

Privilege escalation can take place via weaknesses in the OS. Often a hacker tool is used to exploit a programming flaw or buffer overflow that may allow the attacking user to obtain permanent or temporary access to the administrators group. This form of attack is known as vertical privilege escalation, since the current low-level user or access is itself elevated to a higher level of access. In other cases, privilege escalation occurs through identity theft or credential compromise, such as keystroke capturing or password cracking. This form of attack is known as horizontal privilege escalation, since the attacker switches over to another user account to gain a higher level of access.

Privilege escalation is a violation of security. Specifically, it's a breach of authorization restrictions and may be a breach of authentication. In order to prevent or stop privilege escalation, all OSs should be kept current with patches from the vendor. Additionally, auditing and monitoring should be configured to watch for privilege-escalation symptoms. These include repeated attempts to perform user account management by nonadministrators as well as repeated attempts to access resources beyond a user's assigned authorization level.

ARP poisoning

Address Resolution Protocol (ARP) poisoning is the act of falsifying the IP-to-MAC address resolution system employed by TCP/IP. ARP operates at Layer 2, the Data-Link layer of the OSI model. ARP is responsible for resolving IP addresses into MAC addresses. This allows Layer 2 to physically address transmissions before sending them to the Physical layer (Layer 1). Similar to DNS, ARP resolution is a multistep process:

1. Check the local ARP cache.
2. If that fails, transmit an ARP broadcast.

The ARP broadcast is a transmission to all possible recipients in the local subnet (more accurately, the ARP broadcast is received by all members of the same Ethernet broadcast domain, but that is almost always the same group of systems that is contained in the local subnet), asking all hosts if they own the IP address in question. If the owner of the IP address is present, it responds with a direct reply to the source system with its MAC address.

MAC addresses are essential for TCP/IP communications because transmissions occur from host to host and router to router, based not solely on IP address but primarily on

MAC addresses. When a host sends data to another host, if that host is in the same subnet, it transmits the signal from its MAC-addressed network interface card (NIC) to the target's MAC-addressed NIC. If the target is in a different subnet, it sends the message to the MAC-addressed NIC of the default gateway (which is the router interface in that subnet). Then, that router takes over and tries to find the target host, either with a subnet directly off one of its ports or by sending the message to another router that may have a greater chance of being connected to the target host's subnet. Without proper ARP activity, this process isn't possible.

ARP poisoning can take place in many ways. The most common ways are to poison the local ARP cache or to transmit poisoned ARP replies or announcements. In either case, if a host obtains a false MAC address for an IP address, its transmission is likely to go to the wrong location. This tactic is most effective within a single subnet, but it does have an effect across multiple subnets. ARP poisoning is commonly used in active sniffing attacks where false ARP announcements are used to redirect traffic to the hacker-controlled system, allowing the attacker to view the contents of all transactions. The attack must then forward each Ethernet frame to the correct MAC address destination in order to prevent a DoS and maintain the façade that nothing abnormal is occurring.

One popular tool used to monitor for ARP poisoning is arpwatch. However, the best defense against ARP-based attacks, including ARP poisoning, is port security on the switch. Switch port security can prohibit communications with unknown, unauthorized, rogue devices and may be able to determine which system is responding to all APR queries and block ARP replies from the offending system.

Amplification

In an *amplification attack* the amount of work or traffic generated by an attacker is multiplied in order to cause a significant volume of traffic to be delivered to the primary victim. An amplification attack can also be known as a reflective or bounce attack. Most amplification attacks involve innocent third-party systems or networks, which are used to cause the multiplicity of responses. An historical example of an amplification attack is the Smurf DRDoS discussed earlier. Any attack where a single packet from the attacker generates two or more packets sent to the primary target can be described as an amplification attack. These types of attacks grant the attacker more perceived power and capability than they would have without the multiplication benefit. While the Smurf DRDoS uses the broadcast address of intermediate networks for its amplification, other attacks can use a request, which generates a larger reply.

An example of this latter type of amplification attack was performed in February 2014. The attack took advantage of 4,529 NTP (Network Time Protocol) servers by sending an administrator request of MONLIST. This command generates a report of the last 600 IP addresses of systems requesting information from the NTP server. This caused a flood of traffic against the target victim that reached a peak of 400 Gbps.

Amplification attacks are popular among attackers because they assist in increasing the attack's effective power against larger targets. There have been a few non-amplification attacks that have caused larger levels of traffic against victims—such as the Mirai botnet, which in September 2016 generated 620 Gbps against `krebsonsecurity.com`—but only a few.

DNS poisoning

DNS poisoning is the act of falsifying the DNS information used by a client to reach a desired system. It can take place in many ways. Whenever a client needs to resolve a DNS name into an IP address, it may go through the following process:

1. Check the local cache (which includes content from the HOSTS file).
2. Send a DNS query to a known DNS server.
3. Send a broadcast query to any possible local subnet DNS server. (This step isn't widely supported.)

If the client doesn't obtain a DNS-to-IP resolution from any of these steps, the resolution fails and the communication can't be sent. DNS poisoning can take place at any of these steps, but the easiest way is to corrupt the HOSTS file or the DNS server query.

There are many ways to attack or exploit DNS. An attacker might use one of these techniques:

Deploy a rogue DNS server (also known as DNS spoofing or DNS pharming). A rogue DNS server can listen in on network traffic for any DNS query or specific DNS queries related to a target site. Then the rogue DNS server sends a DNS response to the client with false IP information. This attack requires that the rogue DNS server get its response back to the client before the real DNS server responds. Once the client receives the response from the rogue DNS server, the client closes the DNS query session, which causes the response from the real DNS server to be dropped and ignored as an out-of-session packet.

DNS queries are not authenticated, but they do contain a 16-bit value known as the Query ID or QID. The DNS response must include the same QID as the query to be accepted. Thus, a rogue DNS server must include the requesting QID in the false reply.

Perform DNS poisoning. *DNS poisoning* involves attacking the real DNS server and placing incorrect information into its zone file. This causes the real DNS server to send false data back to clients.

Alter the HOSTS file. Modifying the HOSTS file on the client by placing false DNS data into it redirects users to false locations.

Corrupt the IP configuration. Corrupting the IP configuration can result in a client having a false DNS server definition. This can be accomplished either directly on the client or on the network's DHCP server.

Use proxy falsification. This method works only against web communications. This attack plants false web proxy data into a client's browser, and then the attacker operates the rogue proxy server. A rogue proxy server can modify HTTP traffic packets to reroute requests to whatever site the hacker wishes.

Although there are many DNS poisoning methods, here are some basic security measures you can take that can greatly reduce their threat:

- Limit zone transfers from internal DNS servers to external DNS servers. This is accomplished by blocking inbound TCP port 53 (zone transfer requests) and UDP port 53 (queries).
- Limit the external DNS servers from which internal DNS servers pull zone transfers.

- Deploy a *network intrusion detection system (NIDS)* to watch for abnormal DNS traffic.
- Properly harden all DNS, server, and client systems in your private network.
- Use DNSSEC to secure your DNS infrastructure.
- Require internal clients to resolve all domain names through the internal DNS. This will require that you block outbound UDP port 53 (for queries) while keeping open outbound TCP port 53 (for zone transfers).

Another attack closely related to DNS poisoning and/or DNS spoofing is *DNS pharming*. *Pharming* is the malicious redirection of a valid website's URL or IP address to a fake website that hosts a false version of the original, valid site. This is often part of a phishing attack where the attacker is attempting to trick victims into giving up their logon credentials. If potential victims aren't careful or paying attention, they may be tricked into providing their logon information to the false, pharmed website. Pharming typically occurs either by modifying the local HOSTS file on a system or by poisoning or spoofing DNS resolution. Pharming is an increasingly problematic activity because hackers have discovered means to exploit DNS vulnerabilities to pharm various domain names for large groups of targeted users.

For a detailed review of DNS and its vulnerabilities, read "An Illustrated Guide to the Kaminsky DNS Vulnerability" at `www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html`.

Domain hijacking

Domain hijacking, or domain theft, is the malicious action of changing the registration of a domain name without the authorization of the valid owner. This may be accomplished by stealing the owner's logon credentials; using XSRF, session hijacking, or MitM; or exploiting a flaw in the domain registrar's systems.

Sometimes when another person registers a domain name immediately after the original owner's registration expires this is called domain hijacking, but it should not be. This is a potentially unethical practice, but it is not an actual hack or attack. It is taking advantage of the oversight of the original owner failing to manually extend their registration or configure auto-renewal. If an original owner loses their domain name by failing to maintain registration, there is often no recourse other than to contact the new owner and inquire regarding re-obtaining control. Many registrars have a "you snooze, you lose" policy for lapsed registrations.

When an organization loses their domain and someone else takes over control, this can be a devastating event both to the organization as well as its customers and visitors. The original website or online content will no longer be available (or at least not available on the same domain name). And the new owner might host completely different content or host a false duplicate of the previous site. This later activity might result in fooling visitors, similar to a phishing attack, where PII (personally identifiable information) might be extracted and collected.

Man-in-the-browser

The *man-in-the-browser* (MitB, MiTB, MiB, MIB) attack is effectively an MitM attack. The only real distinction is that the middleman malware is operating on the victim's system, where it is able to intercept and manipulate communications immediately after they leave the browser and before they exit the network interface. Often the MitB is a false proxy system where even encrypted connections can be infiltrated through the presentation of a false, cloned certificate.

The main defenses against MitB attacks are to avoid risky behaviors in order to minimize exposure to malware infection, run an antimalware scanner, use an HIDS, and have a stateful inspection firewall.

LSO (Local Shared Object)

LSOs (local shared objects) are small files or data sets that websites may store on a visitor's computer through the Adobe Flash Player. LSOs, also known as Flash cookies, are generally used to store user preferences and settings, but they do have some risk. LSOs can be used to track a user's web activities and are not cleared or removed when a browser's HTML cookies are cleared.

There are some options to limit the use of LSOs through Adobe Flash configuration settings. However, after each update of Flash, those settings are reset to the default of "Allow". And the most recent versions of Flash will not store LSOs while the browser is operating in privacy or incognito mode.

Malicious Add-ons

Most browsers and many other applications now allow for expansion through downloadable add-ons, *BHOs (browser helper objects)*, *plug-ins,* or *expansion packs*. These add-ons are additional targets for attackers. Hackers have crafted false versions of add-ons, converted add-ons into Trojan horses, and written add-ons to look legitimate but be nothing more than attack code. The purpose is to trick unsuspecting victims into installing the malicious add-ons so the attackers can either gain access to information or take control of the victim's system or identity. Browser add-on stores have started to require signing of add-ons to help address this issue, but it's more important than ever to be cautious about installing anything, to install only software from trusted sources, and to run current antivirus and antimalware scanners.

Header Manipulation

Header manipulation is a form of attack in which malicious content is submitted to a vulnerable application, typically a web browser or web server, under the guise of a valid HTML/HTTP header value. Header manipulation is usually a means to some other nefarious end, such as cross-user defacement, cache poisoning, cross-site scripting, page hijacking, cookie manipulation, open redirects, and so on. In most cases, preventing this attack involves using updated browsers/servers, filtering content from visitors, and rejecting/ignoring any header in violation of HTTP/HTML specifications.

Zero day

Zero-day attacks are newly discovered attacks for which there is no specific defense. A *zero-day exploit* aims to exploit flaws or vulnerabilities in targeted systems that are unknown or undisclosed to the world in general. Zero-day also implies that a direct or specific defense to the attack does not yet exist; thus, most systems with the targeted vulnerable asset are at risk. Another way of describing a zero-day attack is that it is one for which the vendor of the target product has not yet released a patch or update to address the vulnerability; however, there may be IDS or firewall filters that can reduce the risk of attack or exploit while the world waits for the vendor to resolve the concern.

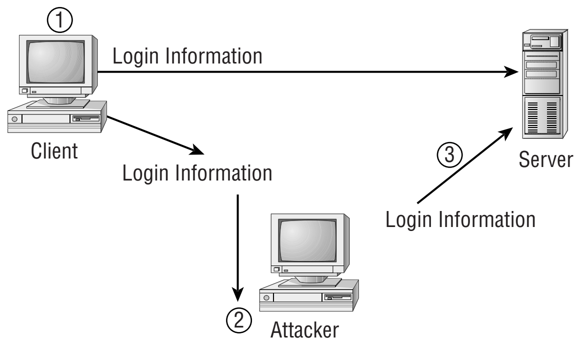
Many attacks take advantage of zero-day vulnerabilities—security flaws discovered by hackers that have not been thoroughly addressed by the security community. There are two main reasons systems are affected by these vulnerabilities. First, it may be the result of the necessary delay between the discovery of a new type of malicious code and the issuance of patches and antivirus updates. Second, it may be due to slowness in applying updates on the part of system administrators. The existence of zero-day vulnerabilities makes it vital that you have a strong patch-management program in your organization that ensures the prompt application of critical security updates. Additionally, you may wish to use a vulnerability scanner to scan your systems for known security issues on a regular basis.

Replay

A *replay attack* is just what it sounds like: an attacker captures network traffic and then replays (retransmits) the captured traffic in an attempt to gain unauthorized access to a system. Most commonly, the attacker focuses on network traffic that is the exchange between a client and server performing authentication. If an attacker can capture the authentication traffic—especially the packets containing the logon credentials, even if they're more than just username and password (such as certificates, token responses, or biometric values)—then a replay attack may grant the attacker the ability to log on to a system by retransmitting the captured packets.

Figure 1.8 shows a replay attack. As the client transmits its logon credentials to the server (1), the attacker intercepts and eavesdrops on that transmission (2) and then later can replay those captured authentication packets against the server to falsify a logon as the original client (3).

FIGURE 1.8 A replay attack occurring



If a replay attack succeeds, the attacker gains the same level of access as the user that originally submitted the authentication information. Fortunately, most modern OSs, networks, protocols, services, and applications use various replay-protection mechanisms to directly prevent such attacks. Common countermeasures are packet sequencing, time stamps, challenge-response, and ephemeral session encryption. Packet sequencing ensures that any packet received that isn't in the proper order (or within a reasonable margin) is dropped and ignored. Packet time stamps ensure that any packet received outside of a specific time window is dropped and ignored. A great example of this is Kerberos, which isn't vulnerable to replay attacks, thanks to its use of time stamps.

Challenge-response is a type of authentication where the server generates and issues a random number challenge to the connecting client. The client uses the challenge number and the hash of the user's password (or other authentication factor) to generate a response. The response is sent back to the server, where it is compared with the expected response generated by the server using the challenge number and the credentials pulled from the user account database. Since each challenge is valid only once and each challenge is randomly selected, replay attacks are not possible.

Ephemeral session key is the term for the use of DHE or ECDHE (see the Chapter 6 section, "Diffie-Hellman") to generate random, nonrepeating, nonreusable, nonpredictable, session-specific symmetric encryption keys. Meeting these criteria means that each authentication session is encrypted, and that encryption is valid only once. Again, it's a reliable method to thwart replay attacks.

Pass the hash

Pass the hash is an authentication attack that potentially can be used to gain access as an authorized user without actually knowing or possessing the plain text of the victim's credentials. This attack is mostly aimed at Windows systems, which maintain a set of cached credentials (this is the item being referenced with the term "hash" in the attack name, which is also known as the authentication token) on client systems for the Windows domains they have authenticated into. The cached credentials are used to grant a user access to the local system and the network in the event the authenticating domain controllers are not available the next time the user attempts to log in. In such a situation, the cached credentials are used, and whenever the domain controllers come back online, the user is automatically accepted by the domain controllers as having been properly authenticated because the user was granted access through the cached credentials from their previous successful domain logon. Although repeated attempts to secure this process have been implemented by Microsoft, hackers continue to exploit this fault-tolerant feature of Windows operating systems.

An attacker extracts the cached credentials from the Registry of a victim's system and then uses those credentials on their own rogue domain client. This may fool the domain controller into accepting the attacker as the authorized user, even though the attack did not actually participate in any authentication process.

Mitigations to this attack include disabling cached credentials, requiring network level authentication, and forcing NTLMv2 (disabling NTLM and LM). Restricted Admin mode is also a good defensive measure. Implementing two-factor authentication can also stop this in some cases.

Hijacking and related attacks

Hijack attacks are those where an attacker takes over control of a session from a valid user. Some forms of hijacking disconnect the client, whereas others grant the attacker a parallel connection into the system or service. This section includes definitions of several hijack-related exploitations.

Clickjacking

Clickjacking is a web page–based attack that causes a user's click to link someplace other than the user intended. This is often accomplished by using hidden or invisible layovers, frame sets, or image maps. When a user sees such an item or link, and then clicks their mouse pointer, the click is intercepted by the invisible or hidden layer, and thus the request is for something other than what the user actually intended.

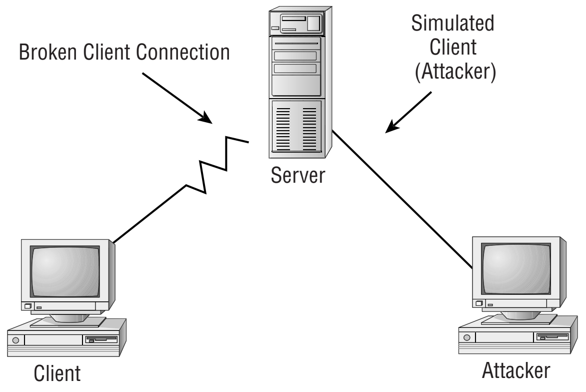
Clickjacking can be used to perform phishing attacks, hijacking, MitM, and MitB. Examples of clickjacking include hiding an Amazon Buy button behind an image of a Play button; tricking users into enabling their microphone or web camera; causing users to set their social media profiles to public; downloading malicious software, including backdoors, rootkits, and ransomware; and falsely generating advertisement clicks.

Session hijacking

TCP/IP hijacking, *or session hijacking*, is a form of attack in which the attacker takes over an existing communication session. The attacker can assume the role of the client or the server, depending on the purpose of the attack. In most forms of session hijacking, the other partner (most often the client) in the communication is disconnected—they're aware that they're no longer communicating and that their session was interrupted. However, they may not immediately realize that they were the collateral damage in a session hijacking attack. Some of the tools that can be used to perform session hijacking are Ettercap, Cain, Juggernaut, and Hunt.

Figure 1.9 shows a TCP/IP hijacking attack.

FIGURE 1.9 TCP/IP hijacking attack



Countermeasures to TCP/IP hijacking attacks include using encrypted protocols and performing periodic, mid-stream reauthentication during a session. Additionally, modern or secured protocols are often designed with preventive features that make session hijacking very difficult or impossible. These features include complex nonlinear sequencing rules as well as time stamps with short timeout values.

URL hijacking

URL hijacking, or *typo squatting*, is a practice employed to capture traffic when a user mistypes the domain name or IP address of an intended resource. A squatter predicts URL typos and then registers those domain names to direct traffic to their own site. This can be done for competition or for malicious intent. The variations used for typo squatting include common misspellings (such as `googel.com`), typing errors (such as `gooogle.com`), variations on a name or word (for example, plurality, as in `googles.com`), and different top-level domains (such as `google.org`).

URL hijacking is also the term applied to the practice of displaying a link or advertisement that looks like that of a well-known product, service, or site, but when clicked redirects the user to an alternate location, service, or product. This may be accomplished by posting sites and pages and exploiting SEO (search engine optimization) to cause your content to occur higher in search results, or through the use of adware that replaces legitimate ads and links with those leading to alternate or malicious locations.

Cookies

A *cookie* is a tracking mechanism developed for web servers to monitor and respond to a user's serial viewing of multiple web pages. A cookie is often used to maintain an e-commerce shopping cart, focus product placement, or track your visiting habits. However, the benign purposes of cookies have been subverted by malevolent entities. Now cookies are a common means of violating your privacy by gathering information about your identity, logon credentials, surfing habits, work habits, and much more.

A cookie can easily be exploited against a web browser to gather sufficient information about a user to allow the attacker to impersonate the victim online. It's generally recommended that you block third-party cookies from everyone and first-party cookies from all but the most trusted sites. Trusted sites are usually those entities that protect your identity by not including such details in a cookie. Instead, these sites only place a session ID in the cookie and keep all of your personal information in a backside database. If you don't allow trusted first-party cookies (aka *session cookies*), functions such as e-commerce shopping carts, online banking, and posting to discussion forums will be disabled.

Cookies can be used in a hijack of a web service connection, where the attacker gains a parallel connection while the original user maintains their connection. This is accomplished by the attacker stealing a copy of the cookie while it's in transit between the valid client and server or directly off the client's storage device. If the cookie serves as an access token, then anyone with possession of it will be recognized by the server as the original authenticated client. The attacker places the cookie on their system and uses their own browser to visit the target server, which mistakenly assumes the attacker is simply another valid connection from the previously authenticated client. Websites should be designed to detect and prevent multiple simultaneous (concurrent) connections.

The Risk of Email Attachments

Because email is so widely used, it has become the most prevalent delivery vehicle for malicious code such as viruses, logic bombs, and Trojan horses. Many of these email-delivered malware items can be used to perform MitM, MitB, or hijacking attacks. To combat this threat, you should deploy an antivirus scanner to scan email content and attachments. You should even consider stripping or blocking email attachments (especially those with known extensions of scripts or executables) as they enter your network (on an email gateway, firewall, and so on). It's always the more secure option to scan, check, and if necessary, strip email on SMTP servers before it reaches an end user's client system.

Typo squatting

See the previous section, "URL Hijacking."

Driver manipulation

Some forms of malicious code or attacker intrusions will take advantage of a form of software manipulation known as *driver manipulation*. Driver manipulation occurs when a malicious programmer crafts a system or device driver so that it behaves differently based on certain conditions. For example, a system benchmark tool may be used to test the performance of a computer, but if the drivers are tuned to provide favorable performance only when the specific benchmarking tool is used, this is an abuse of the evaluation known as driver manipulation. This type of operation occurred recently with Volkswagen, which designed its "fuel-efficient" diesel engines to provide high-performance measurements when being tested but to operate at a lower level during standard driving conditions.

Driver manipulation may be implemented by the original hardware vendor, the original software designer, or a third party, whether a legitimate systems designer or an attacker. Driver manipulation can be based on customized code within the driver itself or on non-driver software that takes advantage of driver features, capabilities, or vulnerabilities in order to achieve the desired goal or effect.

Driver manipulation may be used to achieve a specific goal or hide the fact that a specific goal is not being met. Driver manipulation can be used to optimize performance or diminish performance, improve security or circumvent security, create remote control and backdoor vulnerabilities, or block such abuses from being implemented.

Shimming

Shimming is a means of injecting alternate or compensation code into a system in order to alter its operations without changing the original or existing code. A rough analogy would be that when a table on a new floor is wobbly, a shim can be used to prop up the leg; this is preferable to rebuilding or modifying the table itself. A shim can be used as a quick fix for existing software or firmware code in order to alter operations in situ or to test new options before modifying the core code base.

A shim can be inserted anywhere between two programming objects or subroutines as long as it accepts the output from the preceding element and can produce acceptable input for the receiving element. The shim will intercept the API calls, output, or messages from the first element, perform processing on the captured information set, and then generate output that is compliant with the input of the next element.

Shims are widely used to support legacy applications when the hardware platform no longer provides essential functions. The shim acts as a compatibility interface between the old API and the new one.

Shims can also be employed by attackers to inject alternate commands into an operating environment, add hooks for eavesdropping and manipulation, or simply gain remote access to and control of a target.

Refactoring

Refactoring is a restricting or reorganizing of software code without changing its externally perceived behavior or produced results. Refactoring focuses on improving software's nonfunctional elements, such as quality attributes, nonbehavioral requirements, service requirements, or constraints. Refactoring can improve readability, reduce complexity, ease troubleshooting, and simplify future expansion and extension efforts. Refactoring may be able to simplify internal programmatic logic and eliminate hidden or unresolved bugs or weaknesses.

The goals of refactoring include maintaining the same external behavior and not introducing new bugs or flaws.

Refactoring is about simplifying code, removing redundancies, and avoiding long, monolithic code structures. By dividing computer code into distinct encapsulated elements, modules, objects, or subroutines, programmers ensure that the resulting code is easier to test, verify, and modify. Refactoring is touted by many as a key behavior of experienced programmers.

Refactoring can also be used as a means to focus on programming shortcuts or resolve inelegant solutions. Sometimes, to get code to work, programmers will effectively cheat by using shortcuts rather than crafting the longer valid and complete method. This may be fine initially, but the more elements of the code depend on the cheat, the more unstable and unreliable the whole software becomes. Some call this a technical debt, and like monetary debt, it can accumulate interest and make the resulting software unstable or insecure. Refactoring gives the programmer the opportunity to re-code shortcuts with proper instructions in order to model or craft behaviors more reliably and completely.

The lack of refactoring may leave weaknesses in code or flaws in logic that an attacker might discover and leverage to their advantage. These flaws may be discoverable using fuzzing tools; see the Chapter 3 section "Dynamic analysis (e.g., fuzzing)." Such discoveries are the foundation of unknown and zero-day exploits that anyone using such flawed and inelegant software is likely to be attacked by.

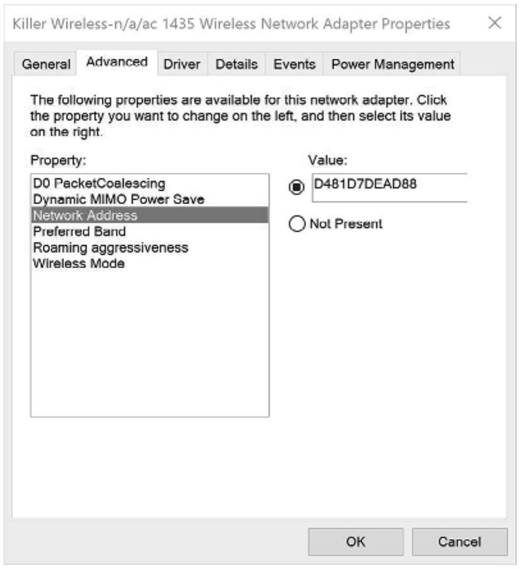
MAC spoofing

MAC (media access control) addresses are also known as physical addresses, hardware addresses, or Ethernet addresses. The MAC address is typically a 48-bit binary number assigned to a NIC by the manufacturer. The MAC address is composed of two equal-sized parts: an OUI and a NIC specific number. The OUI, or organizationally unique identifier, is issued by IEEE (Institute of Electrical and Electronics Engineers) to NIC vendors. Then NIC vendors generate their own NIC-specific number, which may include references to the model and build run as well as a unique value per device. The MAC address is then burned to the ROM chip on the NIC. Because the Ethernet protocol operates in computer memory, however, it reads this ROM-hosted MAC from the NIC but then stores its copy in a software configuration location (such as a CFG file in Linux or the Registry in Windows).

It is possible to eavesdrop on a network and take note of the MAC addresses in use. One of these addresses can then be spoofed into a system by altering the software copy of the NIC's MAC. This causes the Ethernet driver to create frames with the modified or spoofed MAC address instead of the original manufacturer's assigned MAC. Thus, it is quite simple to falsify a MAC address.

MAC spoofing is used to impersonate another system, often a valid or authorized network device, in order to bypass port security or MAC filtering limitations. MAC filtering is a security mechanism intended to limit or restrict network access to those devices with known specific MAC addresses. Its intention is to prevent rogue machines from participating in network communications. However, a simple-to-use Linux application called `macchanger` does just that with a few keystrokes. On the Windows platform, Technitium MAC Address Changer makes MAC spoofing easy. Windows 10 may offer the ability to change your MAC address if supported by the device driver for your NIC; to check, view the Advanced tab of the adapter's device properties dialog box (Figure 1.10). Thus, MAC filtering isn't a complete security solution, as MAC spoofing can bypass this defensive measure.

FIGURE 1.10 Changing a MAC address on a wireless adapter in Windows 10


Killer Wireless-n/a/ac 1435 Wireless Network Adapter Properties
General
Advanced
Driver
Details
Events
Power Management
OK
Cancel

The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:
D0 PacketCoalescing
Dynamic MIMO Power Save
Network Address
Preferred Band
Roaming aggressiveness
Wireless Mode

Value:
[✓ D481D7DEAD88]
Not Present

Countermeasures to MAC spoofing include the following:

- Using intelligent switches that monitor for odd MAC address uses and abuses
- Using a NIDS that monitors for odd MAC address uses and abuses
- Maintaining an inventory of devices and their MAC addresses to confirm whether a device is authorized or unknown and rogue

IP spoofing

Spoofing is the act of falsifying data. Usually the falsification involves changing the source address of network packets. As a result of the changed source address, victims are unable to locate the true attackers or initiators of a communication. Also, by spoofing the source address, the attacker redirects packet responses, replies, and echoes to some other system (as in the case of Smurf, Fraggle, and land DoS attacks).

There are three main types of IP spoofing. One method is to craft IP packets for an attack by setting the source IP address to that of an innocent, uninvolved third party. This type of IP spoofing will result in a simplex or one-way communication for the attacker—any response from the primary victim will be sent to the innocent third party. All logs of the attack event will point to the innocent third-party device as the culprit. A second method is to DoS disconnect the owner/user of an IP address, and then temporarily take on that IP address on the attack system. This provides duplex communication for the attacker in order to retrieve information from the primary target. Once the attack is over, the IP address goes back to normal use by the original system. However, log files blame the attack on the innocent third party who was assigned the IP address, not the rogue system. A third method involves using an IP address from the subnet that is not currently assigned to a valid authorized system. This method also grants the attack duplex communication, but the logs indicate that an unassigned address was the source of the attack, which clearly indicates that a rogue machine must have been using the address to perform the attack.

Countermeasures against IP spoofing attacks include the following:

- Drop all inbound packets received by border systems that have a source destination from inside your private network (this indicates spoofing).
- Drop all outbound packets received by border systems that have a source destination from outside your private network (this also indicates spoofing).
- Drop all packets that have a LAN address in their header if that LAN address isn't officially issued to a valid system.
- Operate a NIDS that monitors for changes in where an IP address is used.

Email Spoofing

Spoofing is also a common activity for unsolicited email, commonly known as spam. Spoofed email means you're unable to reply to the email or determine where it originally came from.

There are innumerable forms of spoofing attacks. Spoofing can be used to redirect packets, bypass traffic filters, steal data, perform social engineering attacks, and even falsify websites.

Countermeasures against email spoofing attacks include using email spam filters as well as the spoofing preventions of IP spoofing.

Wireless attacks

Wireless communication is a quickly expanding field of technologies for networking, connectivity, communication, and data exchange. Literally thousands of protocols, standards, and techniques can be labeled as wireless. These include cell phones, Bluetooth, cordless phones, and wireless networking. As wireless technologies continue to proliferate, your organization's security must go beyond locking down its local network. Security should be an end-to-end solution that addresses all forms, methods, and techniques of communication.

Wireless networking has become common on both corporate and home networks. Properly managing wireless networking for reliable access as well as security isn't always a straightforward proposition. This section examines various wireless security issues.

War Driving

War driving is the act of using a detection tool to look for wireless networking signals. Often, war driving refers to someone looking for wireless networks they aren't authorized to access. In a way, war driving is performing a site survey for possibly malicious or at least unauthorized purposes. The name comes from the legacy attack concept of *war dialing*, which was used to discover active computer modems by dialing all the numbers in a prefix or an area code.

War driving can be performed with a dedicated handheld detector, with a PED (personal electronic device) with WiFi capabilities, or with a notebook that has a wireless network card. It can be performed using native features of the OS, or using specialized scanning and detecting tools.

Once a wireless network is detected, the next step is to determine whether the network is open or closed. An open network has no technical limitations to what devices can connect to it, whereas a closed network has technical limitations to prevent unauthorized connections. If the network is closed, an attacker may try to guess or crack the technologies preventing the connection. Often, the setting making a wireless network closed (or at least hidden) is the disabling of service set identifier (SSID) broadcasting. This restriction is easily overcome with a wireless SSID scanner. After this, the hacker determines whether encryption is being used, what type it is, and whether it can be compromised.

War Chalking

War chalking is a type of geek graffiti that some wireless hackers used during the early years of wireless (1997–2002). It's a way to physically mark an area with information about the presence of a wireless network. A closed circle indicated a closed or secured wireless network, and two back-to-back half circles indicated an open network. War chalking was often used to disclose to others the presence of a wireless network in order to share a discovered Internet link. However, now that Internet connectivity is nearly ubiquitous, with most of us carrying an Internet-connected device on our person (usually a smartphone), the popularity of portable WiFi hotspots, and many retail establishments offering free WiFi as an incentive for customers, the need for and occurrence of war chalking has faded. When an attacker uses war dialing to locate a wireless target to compromise, they don't mark up the area with special symbols to inform others of their intentions.

Replay

A *replay attack* is the retransmission of captured communications in hope of gaining access to the targeted system. This concept was discussed earlier in this chapter in the sections "Application/Service Attacks" and "Replay."

Replay attacks in relation to wireless environments specifically may continue to focus on initial authentication abuse. However, many other wireless replay attack variants exist. They include capturing the new connection request of a typical client, and then replaying that request in order to fool the base station into responding as if another new client connection request had been initiated. Wireless replay attacks can also focus on DoS by retransmitting connection requests or resource requests to the base station in order to keep it busy focusing on managing new connections rather than maintaining and providing service for existing connections.

Wireless replay attacks can be mitigated by keeping the firmware of the base station updated as well as operating a wireless focused NIDS. A W-IDS or W-NIDS will be able to detect such abuses and inform the administrators promptly about the situation.

IV

IV stands for *initialization vector*, a mathematical and cryptographic term for a random number. Most modern crypto functions use IVs in order to increase their security by reducing predictability and repeatability. An IV becomes a point of weakness when it's too short, exchanged in plain text, or selected improperly. Thus, an IV attack is an exploitation of how the IV is handled (or mishandled). One example of an IV attack is that of cracking Wireless Equivalent Privacy (WEP) encryption.

WEP is the original encryption option of 802.11 wireless networking. It's based on RC4. However, because of mistakes in its design and implementation, WEP's primary flaw is related to its IV. The WEP IV is only 24 bits long and is transmitted in plain text. This, coupled with the fact that WEP doesn't check for packet freshness, allows a live WEP crack to be successful in less than 60 seconds (see the Wesside-ng tool from the Aircrack-ng suite at `www.aircrack-ng.org`).

Evil twin

Evil twin is an attack in which a hacker operates a false access point that will automatically clone, or twin, the identity of an access point based on a client device's request to connect. Each time a device successfully connects to a wireless network, it retains a wireless profile in its history. These wireless profiles are used to automatically reconnect to a network whenever the device is in range of the related base station. Each time the wireless adapter is enabled on a device, it wants to connect to a network, so it sends out reconnection requests to each of the networks in its wireless profile history. These reconnect requests include the original base station's MAC address and the network's SSID. The evil twin attack system eavesdrops on the wireless signal for these reconnect requests. Once the evil twin sees a reconnect request, it spoofs its identity with those parameters and offers a plain-text connection to the client. The client accepts the request and establishes a connection with the false evil twin base station. This enables the hacker to eavesdrop on communications through a man-in-the-middle attack, which could lead to session hijacking, data manipulation credential theft, and identity theft.

This attack works because authentication and encryption are managed by the base station, not enforced by the client. Thus, even though the client's wireless profile will include authentication credentials and encryption information, the client will accept whatever type of connection is offered by the base station, including plain text.

To defend against evil twin attacks, pay attention to the wireless network your devices connect to. If you connect to a network that you know is not located nearby, it is a likely sign that you are under attack. Disconnect and go elsewhere for Internet access. You should also prune unnecessary and old wireless profiles from your history list to give attackers fewer options to target.

Rogue AP

A security concern commonly discovered during a site survey is the presence of *rogue wireless access points*. A rogue WAP may be planted by an employee for convenience or it may be operated externally by an attacker.

A wireless access point planted by an employee can be connected to any open network port. Such unauthorized access points usually aren't configured for security or, if they are, aren't configured properly or in line with the organization's approved access points. Rogue wireless access points should be discovered and removed in order to eliminate an unregulated access path into your otherwise secured network.

It's common for an attacker to find a way to visit a company (via a friend who is an employee or by going on a company tour, posing as a repair technician or breakfast taco seller, or even breaking in at night) in order to plant a rogue access point. After a rogue access point is positioned, an attacker can gain entry to the network easily from a modest distance away from your front door.

A rogue WAP can also be deployed by an attacker externally to target your existing wireless clients or future visiting wireless clients. An attack against existing wireless clients requires that the rogue WAP be configured to duplicate the SSID, MAC address, and wireless channel of the valid WAP, although operating at a higher power rating. This may cause clients with saved wireless profiles to inadvertently select or prefer to connect to the rogue WAP instead of the valid original WAP.

The second method focuses on attracting new visiting wireless clients. This type of rogue WAP is configured with a social engineering trick by setting the SSID to an alternate name that appears legitimate or even preferred over the original valid wireless network's SSID. For example, if the original SSID is "ABCcafe," then the rogue WAP SSID could be "ABCcafe-2," "ABCcafe-LTE," or "ABCcafe-VIP." The rogue WAP's MAC address and channel do not need to be clones of the original WAP. These alternate names may seem like better network options to new visitors and thus trick them into electing to connect to the false network instead of the legitimate one.

The defense against rogue WAPs is to be aware of the correct and valid SSID. It would also be beneficial for an organization to operate a wireless IDS to monitor the wireless signals for abuses, such as newly appearing WAPs, especially those operating with mimicked or similar SSID and MAC values.

Jamming

Wireless communications employ radio waves to transmit signals over a distance. There is a finite amount of radio wave spectrum; thus, its use must be managed properly to allow multiple simultaneous connections with little to no interference. The radio spectrum is measured or differentiated using *frequency*. Frequency is a measurement of the number of wave oscillations within a specific time, identified using the unit Hertz (Hz), or oscillations per second. Radio waves have a frequency between 3 Hz and 300 GHz. Different ranges of frequencies have been designated for specific uses, such as AM and FM radio, VHF and UHF television, and so on. Currently, the 900 MHz, 2.4 GHz, and 5 GHz frequencies are the most commonly used in commercial wireless products because of their unlicensed categorization. However, to manage the simultaneous use of the limited radio frequencies, several spectrum-use techniques were developed. These include *spread spectrum, frequency hopping spread spectrum (FHSS)*, *direct sequence spread spectrum (DSSS)*, and *orthogonal frequency-division multiplexing (OFDM)*.

NOTE

Most devices operate within a small subsection of frequencies rather than all available frequencies. This is because of frequency-use regulations (determined by the Federal Communications Commission (FCC) in the United States), power consumption, and the expectation of interference.

Spread spectrum means that communication occurs over multiple frequencies at the same time. Thus, a message is broken into pieces, and each piece is sent at the same time but using a different frequency. Effectively, this is a parallel communication rather than a serial communication.

Frequency hopping spread spectrum (FHSS) was an early implementation of the spread spectrum concept. However, instead of sending data in a parallel fashion, it transmits data in a series while constantly changing the frequency in use. The entire range of available frequencies is employed, but only one frequency at a time is used. As the sender changes from one frequency to the next, the receiver has to follow the same hopping pattern to pick up the signal. FHSS was designed to help minimize interference by constantly shifting frequencies rather than using only a single frequency that could be affected.

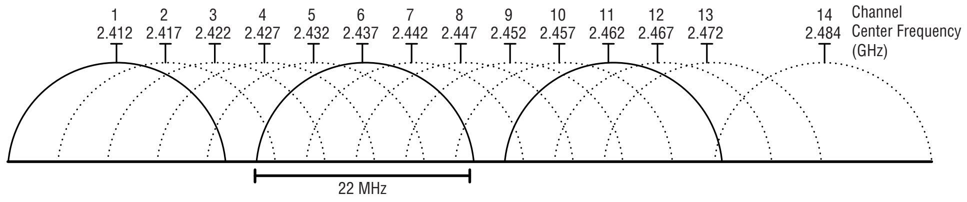
Direct sequence spread spectrum (DSSS) employs several available frequencies simultaneously in parallel. This provides a higher rate of data throughput than FHSS. DSSS also uses a special encoding mechanism known as *chipping code* to allow a receiver to reconstruct data even if parts of the signal were distorted due to interference. This occurs in much the same way that the parity of RAID 5 allows the data on a missing drive to be re-created.

Orthogonal frequency-division multiplexing (OFDM) is yet another variation on frequency use. OFDM employs a digital multicarrier modulation scheme that allows for a more tightly compacted transmission. The modulated signals are perpendicular (orthogonal) and thus don't interfere with each other. Ultimately, OFDM requires a smaller frequency set (aka *channel bands*) but can offer greater data throughput.

Wireless Channels

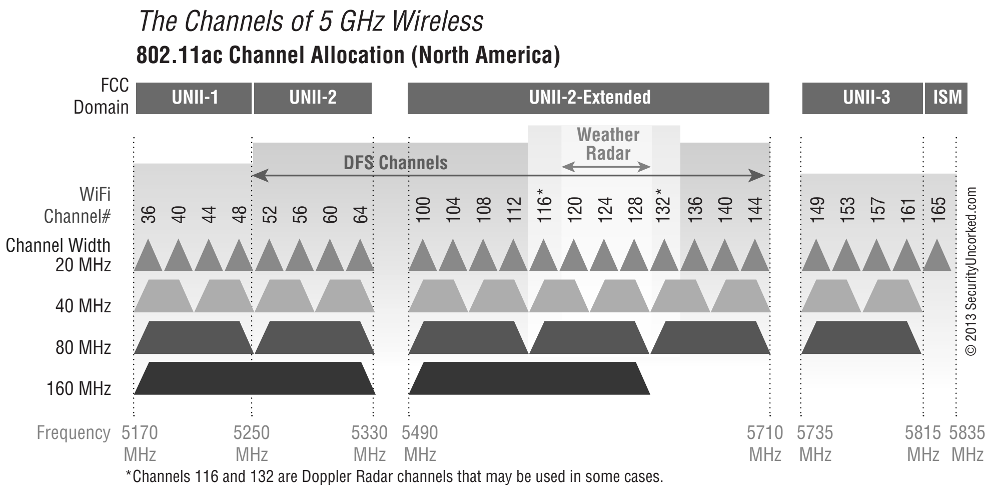
There are many more topics within the scope of wireless networking that we aren't addressing due to space limitations and because they're not covered on the exam. For instance, you may want to learn more about wireless channels. Within the assigned frequency of the wireless signal are subdivisions of that frequency known as *channels*. Think of channels as lanes on the same highway. In relation to the 2.4 GHz frequency, in the United States there are 11 channels, in Europe there are 13, and in Japan there are 14. The differences stem from local laws regarding frequency management (think international versions of the United States' FCC).

The Channels of 2.4 GHz Wireless



Adapted from `https://en.wikipedia.org/wiki/List_of_WLAN_channels`

Wireless communications take place between a client and an access point over a single channel. However, when two or more access points are relatively close to each other physically, signals on one channel can interfere with signals on another channel. One way to avoid this is to set the channels of physically close access points as far apart as possible to minimize channel overlap interference. This is most important for 2.4 GHz networks, where channels are only 5 MHz apart but are 22 MHz wide. This causes channels that are within three numbers of another to experience some level of interference. For example, if a building has four access points arranged in a line along the length of the building, the channel settings could be 1, 11, 1, and 11. But if the building is square and an access point is in each corner, the channel settings may need to be 1, 4, 8, and 11. Think of the signal within a single channel as being like a wide-load truck in a lane on the highway. The wide-load truck is using part of each lane on either side of it, thus making passing in those lanes dangerous. Likewise, wireless signals in adjacent channels will interfere with each other. Channel interference is not an issue with the 5 GHz frequency range (shown next)—the channels are 20 MHz apart and 20 MHz wide, and therefore even adjacent channels do not overlap or interfere.


The Channels of 5 GHz Wireless
802.11ac Channel Allocation (North America)
FCC Domain
UNII-1
UNII-2
UNII-2-Extended
UNII-3
ISM
Weather Radar
DFS Channels
WiFi Channel#
36 40 44 48 52 56 60 64 100 104 108 112 116* 120 124 128 132* 136 140 144 149 153 157 161 165
Channel Width 20 MHz
40 MHz
80 MHz
160 MHz
Frequency 5170 MHz 5250 MHz 5330 MHz 5490 MHz 5710 MHz 5735 MHz 5815 MHz 5835 MHz
*Channels 116 and 132 are Doppler Radar channels that may be used in some cases.
© 2013 SecurityUncorked.com

Adapted from `https://www.networkcomputing.com/wireless/dynamic-frequency-selection-part-3-channel-dilemma/438580919`

Interference may occur by accident or intentionally. Intentional interference is a form of jamming. *Jamming* is the transmission of radio signals to prevent reliable communications by decreasing the effective signal-to-noise ratio. To avoid or minimize interference and jamming, start by adjusting the physical location of devices. Next, check for devices using the same frequency and/or channel. If there are conflicts, change the frequency or channel in use on devices you control. If an interference attack is occurring, try to triangulate the source of the attack and take appropriate steps to address the concern—that is, contact law enforcement if the source of the problem is outside of your physical location.

WPS

WiFi Protected Setup (WPS) is a security standard for wireless networks. It is intended to simplify the effort involved in adding new clients to a well-secured wireless network. It operates by autoconnecting the first new wireless client to seek the network once the administrator has triggered the feature by pressing the WPS button on the base station. However, the standard also calls for a code or PIN that can be sent to the base station remotely in order to trigger WPS negotiation without the need to physically press the button. This can lead to a brute-force guessing attack that could enable a hacker to guess the WPS code in hours (usually less than 6 hours), which in turn enables the hacker to connect their own unauthorized system to the wireless network.

NOTE The PIN code is composed of two four-digit segments, which can be guessed one segment at a time with confirmation from the base station.

WPS is a feature that is enabled by default on most wireless access points because it is a requirement for device WiFi Alliance certification. It's important to disable it as part of a security-focused predeployment process. If a device doesn't offer the ability to turn off WPS (or the Off switch doesn't work), upgrade or replace the base station's firmware or replace the whole device.

Generally, leave WPS turned off. Each time you upgrade your firmware, perform your security-focused predeployment process again to ensure all settings, including WPS, are set properly. If you need to add numerous clients to a network, you can temporarily re-enable WPS—just be sure to disable it immediately afterward.

Bluejacking

Bluejacking involves sending messages to Bluetooth-capable devices without the permission of the owner/user. These messages often appear on a device's screen automatically. Just about any Bluetooth-enabled device, such as a PDA, a cell phone, and even a notebook computer, can receive a bluejacked message. Most bluejacking involves sending a vCard (a virtual business card) to a target device over the Object Exchange (OBEX) protocol (which is also used by infrared communications). Many small portable devices have only a 1 mW power antenna, and Bluetooth may be accessible from 10 meters or less, whereas on a notebook, Bluetooth may be accessible from up to 100 meters (thanks to the 100 mW power antenna). However, even these distances can be exceeded by using a strong transmission antenna, which allows distances of a mile or more.

A bluejack message is often positioned in the name field of the vCard, with little or nothing else. This limits the messages to short strings of text. But this stunt can still be used to pull off various pranks, teasing, and advertisements. Some multimedia message–capable phones are also able to receive images and sound. Bluejacking is mostly harmless, because it doesn't contain malicious code—at least, not so far.

Many devices are configured with a level of defense against bluejacking by not automatically accepting Bluetooth-transmitted messages from unknown (that is, unpaired) sources. Instead, you may see a warning stating that a message from an unknown device has been received and asking whether you want to accept or reject the message. You can also minimize your exposure by keeping Bluetooth off when not in active use.

All Bluetooth devices are vulnerable to bluejacking, since it is just a transmission of a message or announcement. Other Bluetooth-based attacks that are of widespread concern are *bluesniffing* and *bluesmacking*. Bluesniffing is eavesdropping or packet-capturing Bluetooth communications. Since Bluetooth is mostly plain text, this attack can allow an attacker to monitor your Bluetooth activities, such as keystrokes, phone calls, and so on. Bluesmacking is a DoS attack against a Bluetooth device. The defenses for these risks are to minimize use of Bluetooth, especially in public locations.

Bluesnarfing

Bluesnarfing is the unauthorized access of data via a Bluetooth connection. Often the term bluejacking is mistakenly used to describe or label the activity of bluesnarfing. Successful bluesnarfing attacks against PDAs, cell phones, and notebooks have been able to extract calendars, contact lists, text messages, emails, pictures, videos, and more. Because bluesnarfing involves stealing data, it's illegal in most countries.

Bluesnarfing typically occurs over a paired link between the hacker's system and the target device. If the device isn't enabled to be seen by the public (that is, discoverable) or to allow pairing, bluesnarfing usually isn't possible. There was a Bluetooth flaw that could be exploited to perform bluesnarfing against phones that were set up as private, but this has long since been patched. It's true that bluesnarfing is also possible against nondiscoverable devices if you know their Bluetooth MAC addresses, but this usually isn't a practical attack because the 48-bit address must be guessed.

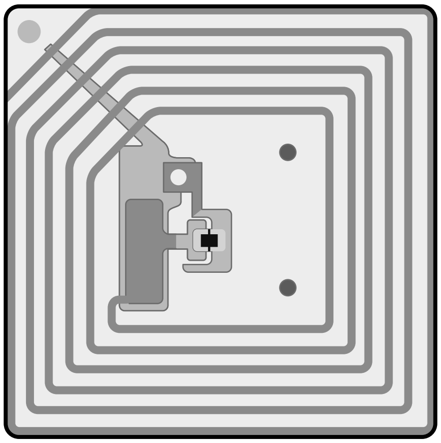
Another interesting Bluetooth attack is *bluebugging*. This attack grants an attacker remote control over the hardware and software of your devices over a Bluetooth connection. The name is derived from enabling the microphone on a compromised system in order to use it as a remote wireless bug.

Bluesnarfing and bluebugging are attacks based not on an inherent flaw of Bluetooth but on vulnerabilities in specific device implementations. Thus, these exploits are not widespread, but they are serious if your device is vulnerable. Be sure to keep your firmware current in order to minimize the risk.

RFID

RFID (Radio Frequency Identification) is a tracking technology based on the ability to power a radio transmitter using current generated in an antenna (Figure 1.11) when placed in a magnetic field. RFID can be triggered/powered and read from a considerable distance away (often hundreds of meters). RFID can be attached to or integrated into the structure of devices such as notebook computers, tablets, routers, switches, USB flash drives, portable hard drives, and so on. This can allow for quick inventory tracking without having to be in direct physical proximity of the device. Simply walking into a room with an RFID reader can collect the information transmitted by the activated chips in the area.

FIGURE 1.11 An RFID antenna



Adapted from `https://electrosome.com/rfid-radio-frequency-identification/`

There is some concern that RFID can be a privacy-violating technology. If you are in possession of a device with an RFID chip, then anyone with an RFID reader can take note of the signal from your chip. Mostly an RFID chip transmits a unique code or serial number—which is meaningless without the corresponding database that links the number to the specific object (or person). However, if you are the only one around and someone detects your RFID chip code, then they can associate you and/or your device with that code for all future detections of the same code.

NFC

Near field communication (NFC) is a standard that establishes radio communications between devices in close proximity. It lets you perform a type of automatic synchronization and association between devices by touching them together or bringing them within inches of each other. NFC is a derivative technology from RFID and is itself a form of field-powered or -triggered device.

NFC is commonly found on smartphones and many mobile device accessories. It's often used to perform device-to-device data exchanges, set up direct communications, or access more complex services such as WPA-2 encrypted wireless networks by linking with the wireless access point via NFC. Because NFC is a radio-based technology, it isn't without its vulnerabilities. NFC attacks can include man-in-the-middle, eavesdropping, data manipulation, and replay attacks.

Disassociation

Disassociation is one of the many types of wireless management frames. A disassociation can be used in several forms of wireless attacks, including the following:

- For networks with hidden SSIDs, a disassociation packet with a MAC address spoofed as that of the WAP is sent to a connected client that causes the client to lose its connection and then send a Reassociation Request packet, which includes the SSID in the clear.
- An attack can send repeated disassociation frames to a client in order to prevent reassociation, thus causing a DoS.
- A session hijack event can be initiated by using disassociation frames to keep the client disconnected while the attacker impersonates the client and takes over their wireless session with the WAP.
- A man-in-the-middle attack can be implemented by using a disassociation frame to disconnect a client. Then the attacker provides a stronger signal from their rogue/fake WAP using the same SSID and MAC as the original WAP; once the client connects to the false WAP, the attacker connects to the valid WAP.

The main defense against these attacks is to operate a wireless IDS, which monitors for wireless abuses.

Cryptographic attacks

Passwords are the most common form of authentication; at the same time, they're the weakest form. Reliance solely on passwords isn't true security. The strength of a password is generally measured in the amount of time and effort involved in breaking the password through various forms of cryptographic attacks. These attacks are collectively known as *password cracking* or *password guessing*. A weak password invariably uses only alphanumeric characters; often employs dictionary or other common words; and may include user profile–related information such as birthdates, Social Security numbers, and pet names. A strong password is longer, more complex, and unique, and is changed on a regular basis.

At least four attack methods are used to steal or crack passwords. All of them involve *reverse hash matching*. This is the process of stealing the hash of a password directly from an authentication server's account database or plucking it out of network traffic, and then reverse-engineering the original password. This is done by taking potential passwords, hashing them, and then comparing the stolen hash with the potential password hash. If a match is found, then the potential password is probably the actual password. (By the way, even if the potential password isn't the actual password, if it happens to produce the same hash, it will be accepted by the authentication system as the valid password.) The four password-cracking or -guessing attacks are *brute force* (aka *birthday attack*), *dictionary, hybrid*, and *rainbow tables*.

This section delves into these four main password-cracking techniques, as well as related issues and a few other attacks that focus on abusing cryptographically protected data.

Birthday

A *brute force or birthday attack* is used against hashing and other forms of cryptography involving finite sets (of either hashes or keys). The birthday attack gets its name from a bar bet that exploits the mathematical probability of shared birthdays. (The bar bet is that you'll drink for free if two people in the bar share a birthday; otherwise, you'll buy the house a round of drinks.) However, the bar bet is derived from the birthday statistical paradox, which is found in the area of mathematics known as *probability theory*.

The issue is that because there are only 366 possible birthdays (don't forget leap year!), the chance of two people sharing the same birth month and day increases exponentially as group size increases. It takes only 23 people for there to be a 50 percent chance that two share the same birthday, and only 75 people are needed for a 99.9 percent chance. When this logic is applied to cracking passwords (or encryption keys), it shows that because the target is part of a finite set (large, yes, but still finite), the likelihood of guessing correctly increases with each subsequent guess. In other words, each wrong guess removes one option from the remaining pool, so the next guess has a slightly greater chance of being correct. This is why brute force attacks are successful—given enough time to perform guesses, the probability of success continues to increase.

Birthday attacks can be waged against any use of hashing. However, they're most commonly employed during password-guessing attacks (discussed in the following section). In a password-guessing attack, a program compares possible passwords with passwords stored in an accounts database. But passwords stored in an accounts database are secured because only their hash values are stored there. Thus, the password-cracking program first performs the same hashing function used by the secured system on each possible password before scanning the accounts database for a match. If a match is found, then the password-guessing tool has discovered a password based on the `f(M)=f(M')` property. This is more specifically known as reverse hash matching. Generally, any form of password cracking is based on the birthday attack.

Known plain text/cipher text

The cryptographic attacks of *known plain text* and *known cipher text* are focused on encryption systems that use the same key repeatedly or that select keys in a sequential or otherwise predictable manner. The goal is to discover the key or a key of the series, and then use that key to determine other keys and thus be able to decrypt most or all of the data protected by the flawed encryption system.

The operation of symmetric encryption involves four main components: the original plain text, the algorithm, the key, and the resultant cipher text. When an attacker knows three of these four parts, they can solve for the final part.

The known plain-text attack starts off by knowing the original data which is to be encrypted. Then the target or victim encrypts the data with the known algorithm with an unknown key. The attacker obtains the cipher text result. From these three parts—plain text, algorithm, and cipher text—the attacker can solve for the key. A slight variant of this is *chosen plain text*, where the attacker provides the victim with the original data, which is then encrypted. Once the key is known, future or past keys can be derived.

The known cipher-text version of this attack is the same process, only in reverse. The attacker knows the cipher text and the algorithm. The victim decrypts the cipher text using the unknown key to produce the plain text, which the attacker obtains. Then the attacker solves for the key. Again, there is a variant of this, known as the *chosen cipher text*, where the attacker provides a data set to the victim to serve as the cipher text, and the attacker retrieves the resulting plain text in order to finally solve for the key.

NOTE In most cases, the cipher text is random and the resulting plain text is unintelligible, but the result is mathematically accurate.

Rainbow tables

Traditionally, password crackers hashed each potential password and then performed an *Exclusive Or (XOR)* comparison to check it against the stolen hash. The hashing process is much slower than the XOR process, so 99.99 percent of the time spent cracking passwords was actually spent generating hashes. A new form of password cracking was developed to remove the hashing time from the cracking time. This technique is known as *rainbow tables*.

Rainbow tables take advantage of a concept known as a *hash chain*. A hash chain is constructed using an initial starting password (often selected at random), and then hashing the starting password into its hash value. Then the hash value is converted into a new password using a process called the *reduction function*. (Note that hashes are not reversible; they are a one-way operation, so the reduction function does not re-create the original password, but a new one.) This process (the password to hash to new password) is repeated numerous times. A hash chain can be composed of just a few links (password to hash to new password sections) or a few thousand. Once crafted, only the starting and finishing passwords for each chain are retained. Many unique hash chains are produced in order to include or cover most or all of the potential passwords and hashes in the range of valid values for the hashing algorithm (password system) being attacked.

Once the rainbow table hash chain database is constructed, it can be used to compromise hashes obtained from a victim. The attacker will first run the stolen hash through the reduction function and then check to see if this value matches any of the hash chain end or stop elements. If so, then the attacker knows the plain text of the password is in that specific chain. If not, the attacker performs another set of hash and reduction functions and checks again. Eventually a matching end of hash chain will be discovered.

Once the correct hash chain is determined, the attacker starts the chain calculation, again starting with the original starting value for the chain, and performs the hash and reduction operations until they encounter the stolen hash. Once that is achieved, the attacker knows the immediately previous password used to produce the hash that matches the stolen hash, and thus the password hash has been cracked.

Although this process may seem complex, or even convoluted, it ends up being a fairly efficient means of compromising passwords. However, rainbow tables do have their limitations. It is difficult to know whether a particular set of hash chains is sufficient to cover or address all or even most of the potential passwords for a given hash. The size of the rainbow table depends on the range of possible passwords, the character options, and the lengths of the passwords. For poor password hashing algorithms—such as LM (LAN Manager), which is the oldest and now depreciated function in Microsoft Windows—a complete rainbow table is only 64 GB in size. A rainbow table covering NTLM passwords—limiting the focus to just U.S. keyboard characters (95 unique options) and passwords with a length of 1–8—would be at least 16 EB (or 16,000,000,000,000,000,000 bytes).

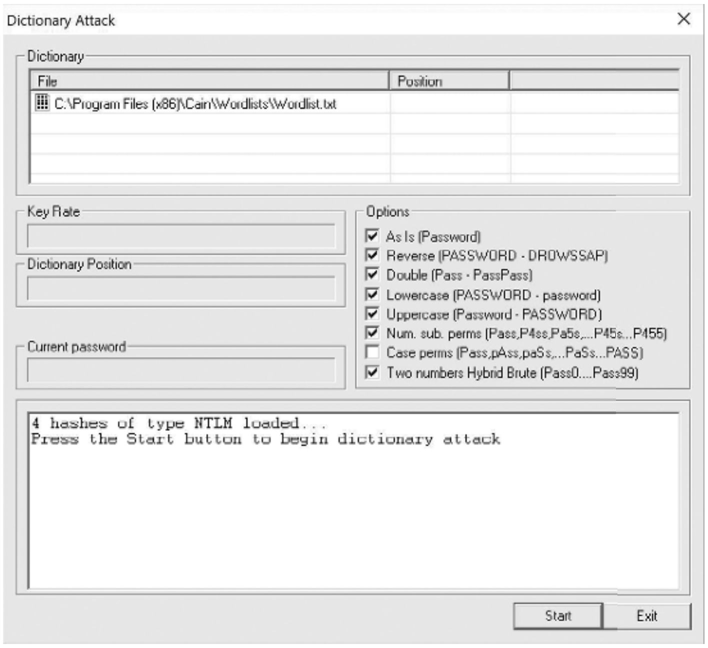
Sometimes rainbow tables are confused with a precomputed hash database. A database containing all possible input passwords and their corresponding output hash would be considerably larger than that of a rainbow table.

To protect yourself from this threat, change all of your passwords to a minimum of 16 characters with a mixture of character types—uppercase, lowercase, numbers, and symbols (when supported). Also be sure to use unique passwords for each logon whether an online site or service or a internal local network system. Whenever available, use multifactor authentication.

Dictionary

A *dictionary attack* (Figure 1.12) performs password guessing by using a preexisting list of possible passwords. Password lists can include millions of possible passwords. Often, password lists or dictionaries are constructed around topics. Thus, if an attacker knows basic information about you as a person, they can attempt to exploit human nature's propensity to select passwords using words common or familiar to you. For example, if an attacker knows that you work in the medical industry, you have cats, and you enjoy sailing, they can select password dictionaries that include words, acronyms, and phrases common to those subjects.

FIGURE 1.12 A dictionary attack configuration page from Cain & Abel



File	Position
C:\Program Files (x86)\Cain\Wordlists\Wordlist.txt	

- [✓] As Is (Password)
- [✓] Reverse (PASSWORD - DROWSSAP)
- [✓] Double (Pass - PassPass)
- [✓] Lowercase (PASSWORD - password)
- [✓] Uppercase (Password - PASSWORD)
- [✓] Num. sub. perms (Pass,P4ss,Pa5s,...P45s...P455)
- [ ] Case perms (Pass,pAss,paSs,...PaSs...PASS)
- [✓] Two numbers Hybrid Brute (Pass0....Pass99)

```
4 hashes of type NTLM loaded...
Press the Start button to begin dictionary attack
```

Dictionary attacks are surprisingly effective against users who haven't been trained in the methods and skills of creating complex passwords. These attacks are fairly simple in that they try only the passwords from the list in the exact form they have in the list. For example, "password" and "Password" and "PASSWORD" are all different, since uppercase and lowercase letters are different ASCII values. Thus, unless all case variants are included in a dictionary list, they would not be tested for by a dictionary attack. Only the specific constructions of passwords included in the dictionary list are used to attack the target password hashes.

Some dictionary lists attempt to include all passwords stolen to date and all valid words (from an actual dictionary, public domain books, or online encyclopedias) in order to cover a broad range of potential passwords and victims' ideas for selecting passwords. One such list is available from `crackstation.net`, where a list of nearly 1.5 billion passwords is available for download. The use of these aggregated password lists is critical for penetration testing and to create a hardened environment. Ethical penetration testers and system administrators should use password-only dictionary lists for security testing, while avoiding lists that include usernames and other personally identifiable information (PII) elements. Dictionary attacks are relatively fast operations, but they have a low rate of success against targets with any knowledge of password security or whose systems enforce reasonable levels of password length and complexity.

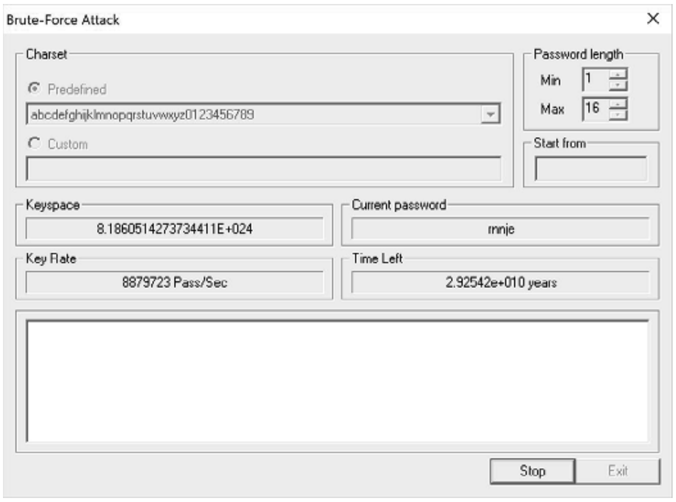
Brute force

A brute force attack (Figure 1.13) is designed to try every possible valid combination of characters to construct possible passwords, starting with single characters and adding characters as it churns through the process, in an attempt to discover the specific passwords used by user accounts. Such attacks are always successful, given enough time. Whereas simple and short passwords can be discovered amazingly quickly with a brute force approach, longer and complex passwords can take an outrageously long period of time (possibly into millions of years of computational time for complex passwords containing 16 or more characters).

Longer and more complex passwords make brute-force attacks less successful. However, given enough time, a brute-force attack will always succeed. But with a sufficiently long target password (16 or more characters), brute force attacks are rendered impractical. An important variant of the brute force attack is that of the hybrid password cracking attack. A *hybrid attack* uses a dictionary list as its password source but uses brute force techniques to make modifications on a progressively increasing level. For example, the first round takes each source password and makes all possible one-character modifications, and then the second round makes all possible two-character modifications, and so on. This includes replacing characters as well as adding characters. The hybrid method has the benefit of focusing on words the target users may have used based on their interests and backgrounds instead of having to try all possible combinations.

Hybrid attacks are often successful even against security professionals who think they're being clever by, for example, changing a to @ and o to 0 and adding the number 12 to the end of the name of their favorite movie character.

FIGURE 1.13 A brute force attack configuration page from Cain & Abel


Brute-Force Attack

Charset
- [✓ Predefined] abcdefghijklmnopqrstuvwxyz0123456789
- Custom

Password length: Min 1, Max 16

Start from

Keyspace: 8.1860514273734411E+024

Current password: mnje

Key Rate: 8879723 Pass/Sec

Time Left: 2.92542e+010 years

Hybrid password attacks are most successful against users who are forced into complying with a company password policy, such as a nine-character minimum length with at least two examples of each of the four character types. Most users are not aware of why password complexity is important, nor does the company training program provide sufficient information on selecting passwords. So, the typical person will seek to adhere to the requirements by selecting a word they can easily remember, and then make modifications until it meets the requirements. Often, workers only work hard enough to be minimally compliant. For example, if a user selects the word “password” as their base word, then with just six alterations to change or add characters, they could produce “P@5sw0Rd!”, which would be in compliance with a nine-character minimum with two examples of each character type password policy. However, this password is extremely poor, because it is based on one of the most likely words to be contained in a dictionary list and is only a six-change/character variant. A typical hybrid attack would find this user’s password in less than 1 second once the base word was reached in the dictionary list.

We must make better password selections in order to defend against hybrid attacks. One method is to select three to five words that are strung together (technically now it would be called a passphrase), and then make sufficient character changes to meet the complexity requirements. This method is far superior to using just a single base word, mostly because the result is usually 10–25 characters long.

Password example	Estimated time to crack
montyp99	minutes
monty python grail	years
monty python movie holy grail	centuries
Monty Python 1975 and the Holy Grail!	millennia

Online vs. offline

An online password attack occurs against a live logon on prompt. In this type of attack, the attacker submits credentials, which are then processed by the authentication service of the target system. If the credentials are correct, then the attacker has successfully impersonated the user. If incorrect, a logon denied error occurs. Most logon prompts offer the user several attempts to provide the correct credentials. However, system managers do not want to grant infinite attempts because that enables hackers to continue attempting to guess the victim's credentials. Thus in most cases, *account lockout* is configured.

Account lockout is the security mechanism that provides a fixed number of logon attempts before the account is locked out (disabled for use). It is common to allow for three logon events before triggering account lockout. Once the lockout is triggered, it can last for a specific number of minutes, such as 15, or indefinitely. If account lockout is set to an indefinite time limit, an administrator will need to manually disable the lockout status in order to return the account to a usable state. Keep in mind that these are just the basics of lockout. There are some forms of lockout that lock the account completely, whereas others lock only the current source attempt location (thus another location or device can be used to attempt logon). Some lockouts will disable the primary means of logon (such as a fingerprint) and revert to a fallback method (such as a password). Some lockout systems also offer a user lockout clearing process (similar to that of password recovery) that may involve SMS or emailed recovery codes or answering identity verification security questions. Some lockout systems use an increasing delay between logon attempts so that the third failure causes a 5-minute delay, the fourth a 15-minute delay, the fifth a 30-minute delay, and so on.

An offline attack is one in which the attacker is not working against a live target system but instead is working on their own independent computers. An attacker will have had to obtain the target's password hashes and then transferred them to their own computers. Collecting hashes is a challenging task, since most systems are designed to specifically prevent theft of hashes. They are stored in the encrypted authentication database and only sent over network communications via encrypted channels. Thus, an attacker must use clever techniques to access the hashes. These can include direct physical contact with the target system, using a remote control tool to extract the hashes from memory or from network traffic, or obtaining access to a backup of the system files.

Once the attacker has the password hashes, password-cracking operations can take place on the attacker's own computers, either on their CPU or GPU, or using a cloud computing service (such as Amazon's EC2 service). An offline attack is not affected or limited by account lockout, since that security feature is part of the authentication service and not the hash itself. So the attacker has no limit to the number of password-cracking events to attempt other than their own system's computation speed and their patience to allow the attack to operate.

Collision

A *collision* occurs when the output of two cryptographic operations produce the same result. Collisions occur in relation to encryption operations as well as hashing operations (see Chapter 6, "Cryptography and PKI," for the full coverage of cryptography concepts).

When two encryption operations produce the same cipher text, a collision has occurred. When collisions occur in relation to symmetric encryption, it is a symptom of a serious problem. It can mean that the encryption system being used is not properly implementing the algorithm or that its use of randomization is flawed. A secure encryption system will guarantee that, even if you attempt to encrypt the same message a second time using the same algorithm and the same key, the randomization function (known as the initialization vector [IV]) will ensure unique cipher text each time. An encryption collision can also indicate that the encryption key is not being used in an ephemeral manner (that is, once, randomly, and in a nonrepeating way).

Hashing collisions are more likely the focus of discussion or concern. A hash collision occurs when two different data sets that are hashed by the same hashing algorithm produce the same hash value. This is always a possibility with hashing, since it is a process that takes any input to produce a fixed-length output. Thus, there is a guarantee that multiple inputs will produce the same output. The key is to understand how to use hashing properly in order to avoid allowing occurrences of collisions to be an actual violation of integrity.

Hashes are designed to provide protection from corruption, alteration, or counterfeiting that a person would not notice or would overlook. To this end, hashing algorithms employ features known as *avalanche effects*, which ensure that small changes in the input produce large changes in the output. Thus, if before and after hashes do not match, the current data set is not the same as the original data set. However, if the before and after hashes do match, then there is still one additional step before the current data can be accepted as being the unchanged original. That step is to look at the data, and if it does not look like the original, then the occurrence of a matching hash is a collision. This is therefore a situation where the data has been switched out with a different data set that happens to produce the same hash. In this case, the data should still be discarded because it is obviously not the original. Only when the before and after hashes match and the current data looks like the original data can you accept the current data as valid and unchanged (that is, it has retained its integrity).

Hash collision attacks are intended to fool a victim into accepting an alternate data set just because it happens to produce the same hash value. This can occur only if the victim does not look at their data in addition to checking the hash values. Hash collision is easier with hashes that are shorter, such as 128 bits in length, than longer hashes, such as those 512 bits in length. So when choosing a hashing algorithm, use the available option that produces the longest hash.

Downgrade

A *downgrade attack* attempts to prevent a client from successfully negotiating robust high-grade encryption with a server. This attack may be performed using a real-time traffic manipulation technique or through a man-in-the-middle attack (a false proxy) in order to forcibly downgrade the attempted negotiation to a lower quality level of algorithms and key exchange/generation. By keeping the client from setting up a high-grade encrypted session, the attacker is able to continue to eavesdrop and manipulate the conversation even after the "encrypted" session is established. This type of attack is possible if both the client and server retain older encryption options designed for backward compatibility. If the attacker can force the negotiation to select these older options, then the attacker may be able to exploit known weaknesses in the older solutions. One example of the downgrade attack is against SSL/TLS, where the attacker uses a technique known as the *POODLE attack*. POODLE stands for Padding Oracle On Downgraded Legacy Encryption. POODLE causes the client to fall back to using SSL 3.0, which has less robust encryption cipher suite options than TLS.

The best defense against downgrade attacks is to disable support for older encryption options and backward compatibility with less secure systems.

Replay

A *replay attack* occurs when an attacker captures network packets and then retransmits or replays them back onto the network. Often a replay attack focuses on authentication packets with the goal of being falsely granted access to a system or service. In a replay (or playback) attack, the attacker does not gain knowledge of the victim's credentials. Instead, the attacker holds the packets that contain the credentials and then sends them back out onto the network in hopes of fooling the authentication service into granting the attacker the same access as the original user (the victim).

Replay attacks are mostly relegated to legacy systems and services because most modern implementations of authentication have specific defenses against replay attacks integrated into them. Such defenses include using short time stamps in packets so they are valid only for a few moments, using random one-time-use challenge-response dialogs (that is, a random number is sent to the client system, which must calculate a response), and using one-time-use ephemeral session encryption keys.

Weak Implementations

Most failures of modern cryptography systems are due to poor or *weak implementations* rather than a true failure of the algorithm itself. The algorithms employed by most software products have been widely scrutinized and have survived focused analyses and attacks. Thus, any failures of the software products to provide reliable security are due to programming mistakes or errors in implementation.

It is challenging to properly implement randomization functions in software that produce truly random and non-predictable keys. Some programmers ignore this complexity (and it's important) and rely on the simple and predictable "randomization" function provided by their execution environment. Other failures include reusing the key across multiple data sets or sessions, using keys in sequential order or other patterns of use, storing keys in an insecure fashion, and distributing or exchanging keys in an insecure manner. All of these issues have resolutions, mostly adopting secure coding practices and consulting with other expert programs for tips and suggestions on avoiding common programming pitfalls. It would also be helpful to submit new code to thorough review and analysis by other skilled programmers and cryptography subject matter experts (SMEs).

Exam Essentials

Understand social engineering. Social engineering is a form of attack that exploits human nature and human behavior. Social engineering attacks take two primary forms: convincing someone to perform an unauthorized operation or convincing them to reveal confidential information.

Understand phishing. Phishing is the process of attempting to obtain sensitive information such as usernames, passwords, credit card details, or other personally identifiable information (PII) by masquerading as a trustworthy entity (a bank, a service provider, or a merchant, for example) in electronic communication (usually email).

Understand spear phishing. Spear phishing is a more targeted form of phishing where the message is crafted and directed specifically to an individual or group of individuals. The hope of the attack is that someone who already has an online/digital relationship with an organization is more likely to fall for the false communication.

Understand whaling. Whaling is a form of phishing that targets specific high-value individuals.

Understand vishing. Vishing is phishing done over VoIP services.

Understand tailgating and piggybacking. Tailgating occurs when an unauthorized entity gains access to a facility under the authorization of a valid worker but without their knowledge. Piggybacking occurs when an unauthorized entity gains access to a facility under the authorization of a valid worker but with their knowledge and consent.

Understand impersonation. Impersonation is the act of taking on the identity of someone else. The purpose of impersonation is to trick someone into believing you're the claimed identity so you can use the power or authority of that identity. Impersonation is also known as masquerading or spoofing.

Understand dumpster diving. Dumpster diving is the act of digging through trash in order to obtain information about a target organization or individual. It can provide an attacker with information that could make social engineering attacks easier or more effective.

Understand shoulder surfing. Shoulder surfing occurs when someone is able to watch your keyboard or view your display. This may allow them to learn your password or see information that is confidential, private, or simply not for their eyes.

Understand hoaxes. A hoax is a form of social engineering designed to convince targets to perform an action that will cause problems or reduce their IT security. A hoax is often an email that proclaims some imminent threat is spreading across the Internet and that you must perform certain tasks in order to protect yourself.

Understand watering hole attacks. A watering hole attack is a form of targeted attack against a region, a group, or an organization. It's waged by poisoning a commonly accessed resource.

Understand principles of social engineering. Many techniques are involved in social engineering attacks. These often involve one or more common principles such as authority, intimidation, consensus/social proof, scarcity, familiarity/liking, trust, and urgency.

Understand arbitrary code execution. Arbitrary code execution is the ability to run any software on a target system.

Understand DoS. Denial of service (DoS) is a form of attack that has the primary goal of preventing the victimized system from performing legitimate activity or responding to legitimate traffic. One form exploits a weakness, an error, or a standard feature of software to cause a system to hang, freeze, consume all system resources, and so on. The end result is that the victimized computer is unable to process any legitimate tasks. Another form floods the victim's communication pipeline with garbage network traffic. The end result is that the victimized computer is unable to send or receive legitimate network communications.

Understand a Smurf attack. This form of DRDoS uses ICMP echo reply packets (ping packets).

Understand Xmas attacks. The Xmas attack is actually an Xmas scan. It's a form of port scanning that can be performed by a wide number of common port scanners, including Nmap, Xprobe, and hping2. The Xmas scan sends a TCP packet to a target port with the flags URG, PSH, and FIN all turned on.

Understand DDoS. Distributed denial-of-service (DDoS) employs an amplification or bounce network that is an unwilling or unknowing participant that is unfortunately able to receive broadcast messages and create message responses, echoes, or bounces. In effect, the attacker sends spoofed message packets to the amplification network's broadcast address.

Understand man-in-the-middle attacks. A man-in-the-middle attack is a form of communications eavesdropping attack. Attackers position themselves in the communication stream between a client and server (or any two communicating entities). The client and server believe they're communicating directly with each other.

Understand buffer overflows. Buffer overflows occur due to a lack of secure defensive programming. The exploitation of a buffer overflow can result in a system crash or arbitrary code execution. A buffer overflow occurs when a program receives input that is larger than it was designed to accept or process. The extra data received by the program is shunted over to the CPU without any security restrictions; it's then allowed to execute. Results of buffer overflows can include crashing a program, freezing or crashing the system, opening a port, disabling a service, creating a user account, elevating the privileges of an existing user account, accessing a website, or executing a utility.

Understand injection attacks. An injection attack is any exploitation that allows an attacker to submit code to a target system in order to modify its operations and/or poison and corrupt its data set. Examples include SQL injection, LDAP injection, XML injection, command injection, HTML injection, code injection, and file injection.

Understand SQL injection. SQL injection attacks allow a malicious individual to perform SQL transactions directly against the underlying database through a website front end.

Understand directory traversal. A directory traversal is an attack that enables an attacker to jump out of the web root directory structure and into any other part of the filesystem hosted by the web server's host OS.

Understand cross-site scripting. Cross-site scripting (XSS) is a form of malicious code injection attack in which an attacker is able to compromise a web server and inject their own malicious code into the content sent to other visitors.

Understand cross-site scripting (XSS) prevention. The most effective ways to prevent XSS on a resource host are implemented by the programmer by validating input, coding defensively, escaping metacharacters, and rejecting all script-like input.

Understand cross-site request forgery (XSRF). Cross-site request forgery (XSRF) is an attack focused on the visiting user's web browser more than on the website being visited. The main purpose of XSRF is to trick the user or the user's browser into performing actions they had not intended or would not have authorized.

Understand cross-site request forgery (XSRF) prevention. XSRF prevention measures include adding a randomization string (called a nonce) to each URL request and session establishment and checking the client HTTP request header referrer for spoofing.

Understand privilege escalation. Privilege escalation occurs when a user account is able to obtain unauthorized access to higher levels of privileges, such as a normal user account that can perform administrative functions. Privilege escalation can occur through the use of a hacker tool or when an environment is incorrectly configured.

Understand ARP poisoning. ARP poisoning is the act of falsifying the IP-to-MAC address resolution system employed by TCP/IP.

Understand amplification. An amplification attack is one where the amount of work or traffic generated by an attacker is multiplied in order to cause a significant volume of traffic to be delivered to the primary victim. An amplification attack can also be known as a reflective or bound attack.

Understand DNS poisoning. DNS poisoning is the act of falsifying the DNS information used by a client to reach a desired system. This can be accomplished by deploying a rogue DNS server (also known as DNS spoofing and DNS pharming), using DNS poisoning, altering the HOSTS file, corrupting IP configuration, and using proxy falsification.

Understand pharming. Pharming is the malicious redirection of a valid website's URL or IP address to a fake website that hosts a false version of the original valid site.

Understand domain hijacking. Domain hijacking or domain theft is the malicious action of changing the registration of a domain name without the authorization of the valid owner. This may be accomplished by stealing the owner's logon credentials, using XSRF, hijacking sessions, using MitM, or exploiting a flaw in the domain registrar's systems.

Understand man-in-the-browser. The man-in-the-browser (MitB, MiTB, MiB, MIB) attack is effectively a MitM attack. The only real distinction is that the middle-man malware is operating on the victim's system, where it is able to intercept and manipulate communications immediately after they leave the browser and before they exit the network interface.

Understand zero day. Zero-day attacks are newly discovered attacks for which there is no specific defense. A zero-day exploit aims at exploiting flaws or vulnerabilities in targeted systems that are unknown or undisclosed to the world in general. Zero day also implies that a direct or specific defense to the attack does not yet exist; thus most systems with the targeted vulnerable asset are at risk.

Understand a replay attack. In a replay attack, an attacker captures network traffic and then replays the captured traffic in an attempt to gain unauthorized access to a system.

Understand pass the hash. Pass the hash is an authentication attack that potentially can be used to gain access as an authorized user without actually knowing or possessing the plain text of the victim's credentials. This attack is mostly aimed at Windows systems.

Understand hijacking attacks. Hijacking attacks are those where an attacker takes over control of a session from a valid user. Some forms of hijacking disconnect the client, whereas others grant the attacker a parallel connection into the system or service.

Understand clickjacking. Clickjacking is a web page–based attack that causes a user to click on something other than what the user intended to click. This is often accomplished by using hidden or invisible layovers, frame sets, or image maps.

Understand session hijacking. TCP/IP hijacking, or session hijacking, is a form of attack in which the attacker takes over an existing communication session. The attacker can assume the role of the client or the server, depending on the purpose of the attack.

Understand typo squatting/URL hijacking. Typo squatting, or URL hijacking, is a practice employed to capture traffic when a user mistypes the domain name or IP address of an intended resource.

Understand cookies. A cookie is a tracking mechanism developed for web servers to monitor and respond to a user's serial viewing of multiple web pages. It may allow identity theft.

Understand driver manipulation. Driver manipulation occurs when a malicious programmer crafts a system or device driver so that it behaves differently based on certain conditions.

Understand shimming. Shimming is a means of injecting alternate or compensation code into a system in order to alter its operations without changing the original or existing code.

Understand refactoring. Refactoring is a restricting or reorganizing of software code without changing its externally perceived behavior or produced results. Refactoring focuses on improving software's nonfunctional elements, such as quality attributes, non-behavioral requirements, service requirements, and constraints.

Understand spoofing. Spoofing is the act of falsifying data. Usually the falsification involves changing the source addresses of network packets. Because the source address is changed, victims are unable to locate the true attackers or initiators of a communication. Also, by spoofing the source address, attackers redirect responses, replies, and echoes of packets to some other system.

Understand MAC spoofing. MAC spoofing is used to impersonate another system, often a valid or authorized network device in order to bypass port security or MAC filtering limitations.

Understand IP spoofing. There are three main types of IP spoofing: crafting IP packets for an attack but setting the source IP address to that of an innocent, uninvolved third party; via DoS, disconnecting the owner/user of an IP address, then temporary taking on that IP address on the attack system; or using an IP address from the subnet that is not currently assigned to a valid authorized system.

Understand war driving. War driving is the act of using a detection tool to look for wireless networking signals. Often, war driving is the process of someone looking for a wireless network they aren't authorized to access.

Understand wireless replay attacks. Wireless replay attacks may focus on initial authentication abuse. They may be used to simulate numerous new clients or cause a DoS.

Understand initialization vector (IV). IV is a mathematical and cryptographic term for a random number. Most modern crypto functions use IVs in order to increase their security by reducing predictability and repeatability.

Understand evil twin attacks. During an evil twin attack, a hacker configures their system as a twin of a valid wireless access point. Victims are tricked into connecting to the fake twin instead of the valid original wireless network.

Understand rogue access points. A rogue WAP may be planted by an employee for convenience or it may be operated externally by an attacker. Rogue wireless access points should be discovered and removed in order to eliminate an unregulated access path into your otherwise secured network.

Understand jamming. Jamming is the transmission of radio signals to prevent reliable communications by decreasing the effective signal-to-noise ratio.

Understand WPS attacks. WPS is a security standard for wireless networks that was found to be flawed. The standard called for a code that could be sent to the base station remotely in order to trigger WPS negotiation. This led to a brute force guessing attack that could enable a hacker to guess the WPS code in just hours.

Understand bluejacking. Bluejacking is the sending of messages to Bluetooth-capable devices without the permission of the owner/user. Just about any Bluetooth-enabled device, such as a smartphone or notebook computer, can receive a bluejacked message.

Understand bluesnarfing. Bluesnarfing is the unauthorized accessing of data via a Bluetooth connection. Successful bluesnarfing attacks against smartphones and notebooks have been able to extract calendars, contact lists, text messages, emails, pictures, videos, and more.

Understand RFID. RFID (radio frequency identification) is a tracking technology based on the ability to power a radio transmitter using current generated in an antenna when placed in a magnetic field. RFID can be triggered/powered and read from up to hundreds of meters away.

Understand NFC. Near field communication (NFC) is a standard to establish radio communications between devices in close proximity. It lets you perform a type of automatic synchronization and association between devices by touching them together or bringing them within inches of each other.

Understand disassociation. Disassociation is one of the many types of wireless management frames. A disassociation can be used in several forms of wireless attacks, including discovering hidden SSIDs, causing a DoS, hijacking sessions, and using MitM.

Understand password attacks. The strength of a password is generally measured in the amount of time and effort involved in breaking the password through various forms of cryptographic attacks. These attacks are collectively known as password cracking or password guessing. Forms of password attacks include brute force (also known as a birthday attack), dictionary, hybrid, and rainbow tables.

Understand password guessing. Password guessing is an attack aimed at discovering the passwords employed by user accounts. It's often called password cracking. There are two primary categories of password-guessing tools based on the method used to select possible passwords for a direct logon prompt or birthday attack procedure: brute force and dictionary.

Understand password crackers. A password cracker is a tool used to reverse-engineer the secured storage of passwords in order to gain (or regain) access to an unknown or forgotten password. There are four well-known types of password-cracking techniques: dictionary, brute force, hybrid, and precomputed hash.

Understand birthday attacks. The birthday attack exploits a mathematical property that if the same mathematical function is performed on two values and the result is the same, then the original values are the same. This concept is often represented with the syntax `f(M)=f(M') therefore M=M'`.

Understand known plain text and known cipher text attacks. The cryptographic attacks of known plain text and known cipher text are focused on encryption systems that use the same key repeatedly or that select keys in a sequential or otherwise predictable manner. The goal is to discover the key or a key of the series, and then use that key to determine other keys and thus be able to decrypt most or all of the data protected by the flawed encryption system.

Understand rainbow tables. Rainbow tables take advantage of a concept known as a hash chain. It offers relatively fast password cracking, but at the expense of spending the time and effort beforehand to craft the rainbow table hash chain database.

Understand dictionary attacks. A dictionary attack performs password guessing by using a preexisting list of possible passwords.

Understand brute-force attacks. A brute force attack is designed to try every valid combination of characters to construct possible passwords, starting with single characters and adding characters as it churns through the process, in an attempt to discover the specific passwords used by user accounts.

Understand online vs. offline password cracking. An online password attack occurs against a live logon prompt. An offline attack is one where the attacker is not working against a live target system, but instead is working on their own independent computers to compromise a password hash.

Understand collision. A collision is when the output of two cryptography operations produces the same result. Collisions occur in relation to encryption operations as well as hashing operations.

Understand a downgrade attack. A downgrade attack attempts to prevent a client from successfully negotiating robust high-grade encryption with a server. This attack may be performed using a real-time traffic manipulation technique or through a man-in-the-middle attack (a false proxy) in order to forcibly downgrade the attempted negotiation to a lower quality level of algorithms and key exchange/generation.

Understand replay attacks. A replay attack is one in which an attacker captures network packets and then retransmits or replays them back onto the network.

Understand weak implementations. Most failures of modern cryptography systems are due to poor or weak implementations rather than a true failure of the algorithm itself.

1.3 Explain threat actor types and attributes.

A *threat actor* is the person or entity who is responsible for causing or controlling any security-violating incidents experienced by an organization or individual. Such incidents may or may not successfully breach the security infrastructure of the victim, but any attempt is still an event to be noticed, recorded, and evaluated. It is important to understand the threats faced by your organization, and knowing the types of threat actors who may be responsible will further help your preparedness efforts.

Types of actors

The actual perpetrators of attacks or exploits range from individuals to organizations. Some of the terms related to threat actors are included in the following sections.

Black Hat, White Hat, Gray Hat

There are many names that have been used to refer to those who attack computer systems and networks. These include hacker, cracker, phreaker, black hat, white hat, and gray hat. A hacker is someone skilled and knowledgeable in a system. Hackers may be able to take a system apart, alter its functions, repair broken elements, and reassemble it back into a working system. The term *hacker* simply denotes skill, not intention or authorization. A cracker is an attacker of computer systems and networks. It is the malicious form of hacker. However, due to media use, the term *hacker* has picked up a negative connotation. So, *ethical hacker* is often used to denote the benign nature of the skilled individual, versus *criminal or malicious hacker* for the bad guy. A *phreaker* is someone who attacks the telephone network and related systems. A *black hat* is a criminal or malicious attacker, whereas a *white hat* is an ethical hacker or skilled IT professional. A *gray hat* may be a reformed criminal or a skilled IT professional operating undercover to perform ethical hacking (also known as penetration testing).

Script kiddies

Script kiddies are threat actors who are less knowledgeable than a professional skilled attacker. A script kiddie is usually unable to program their own attack tools and may not understand exactly how the attack operates. However, a script kiddie is able to follow instructions and use attack tools crafted by other skilled and knowledgeable malicious programmers. Script kiddies are much more numerous than professional attackers. Script kiddies also pose a serious threat due to their number as well as the chance that they may have access to an attack tool that can exploit a vulnerability in your IT system.

Hacktivist

A *hacktivist* is someone who uses their hacking skills for a cause or purpose. A hacktivist commits criminal activities to further their cause. A hacktivist attacks targets even when they know they will be identified, apprehended, and prosecuted. They do this because they believe their purpose or cause is more important than themselves. Keep in mind that committing crimes is still illegal, no matter what the intention or purpose of the perpetrator is. Hacktivism may often be used as a form of protest, but it is not a legal one.

Organized crime

Organized crime is involved in cybercrime activities because it is yet another area of exploitation that may allow criminals to gain access, power, or money. Although not all hacks and attacks are funded or backed by organized crime groups, their involvement in cyberattacks is quite significant. Some organized crime syndicates actively recruit skilled hackers to join their ranks in the criminal enterprise.

Nation states/APT

Many governments and militaries—nation-states—are now using cyberattacks as yet another weapon in their arsenal against real or perceived enemies, whether internal or outside their borders. For examples of nation-state–sponsored cyber events, read up on the malicious code concepts of Stuxnet (uncovered by Symantec) (`https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`) and Flame (uncovered by Kaspersky) (`https://www.kaspersky.com/flame`).

APT (advanced persistent threat) is any form of cyberattack that is able to continually exploit a target over a considerable period of time. An APT often takes advantage of unknown flaws (that is, not publicly known) and tries to maintain stealth throughout the attack. The name is derived from the concept that the attacks are unique and exploit flaws that are not public knowledge (that is, this state of using an exploit against an unknown flaw is labeled as advanced), that the exploit grants the attackers ongoing remote access to and control over the target (that is, it's persistent), and that the attackers are likely nation-states (that is, it's a threat). Sometimes APT is used to refer to the threat actors who continue to focus on targets using all available exploitations in order to retain control and dominance over the victim.

Insiders

One of the biggest risks at any organization is its own internal personnel. Hackers work hard to gain what insiders already have: physical presence within the facility or a working user account on the IT infrastructure. When an insider performs malicious activities, the threat is significant, because they're already past most physical barriers and may have easy access that lets them compromise IT security.

Malicious insiders can bring in malicious code from outside on various storage devices, including smartphones, memory cards, optical discs, and USB drives. These same storage devices can be used to leak or steal internal confidential and private data in order to disclose it to the outside world. (Where do you think most of the content on WikiLeaks comes from?) Malicious insiders can execute malicious code, visit dangerous websites, or intentionally perform harmful activities.

The means to reduce the threat of malicious insiders include thorough background checks, strong policies with severe penalties, detailed user activity auditing and monitoring, prohibition of external and private storage devices, and use of whitelists to minimize unauthorized code execution.

Competitors

Another type of threat actor is that of competitors. Many organizations still elect to perform corporate espionage and sabotage against their competition while it is widely known that such actions are illegal. This might be due to a perceived advantage another company has or the lack of desire to put forth the time and effort to gain valid market share in a competitive marketplace. Organizations should always take care to closely monitor their competition for signs that they are benefiting from and launching cyberattacks. This concept is known as competitive intelligence gathering. Competitive intelligence gathering is a valid and legal means to keep track of another company by analyzing publicly available information.

Companies should also pay special attention to business partners, contractors, and employees who may have left an organization only to gain employment with a competitor (whether you hire someone from the other firm or they hire away your employees).

Attributes of actors

Threat actors can have a wide range of skills and attributes. When analyzing the threats to your organization, it is important to keep these variables in mind.

Internal/external

Threats can originate from inside your organization as well as outside. All too often, companies focus most of their analysis and security deployment efforts on external threats without providing sufficient attention to the threats originating from inside. All threats should be considered on their merits—their specific risk level to your organization and its assets—and not just based on someone's subjective perspective on the issues.

Level of sophistication

Threat actors can vary greatly in their skill level and level of sophistication. Some attackers are highly trained professionals who are applying their education to malicious activities, whereas others are simply bad guys who learned how to perform cyberattacks just to expand their existing repertoire.

Some attacks are structured or targeted, others are unstructured and opportunistic. A structured or targeted attack is one where a specific organization was always the focus and considerable effort was expended to find a means to compromise that organization's security. This type of an attack usually involves a higher level of sophistication because there is a need to be methodical and persistent in seeking to accomplish the goal. An unstructured or opportunistic attack is one that seeks out a target that happens to be vulnerable to a chosen attack or exploit. This type of attack is often performed by an attacker once they have crafted a new exploit tool; they seek out a target to show off or demonstrate that their tool actually works. This type of attack displays a much lower level of sophistication. Yes, building a new attack tool can be a complex process, but the action of hitting multiple targets until you finally locate one with a particular weakness is not complicated. This is the equivalent of giving a toddler a hammer; if they treat everything like a nail and hit it, maybe eventually they will actually hit a real nail.

Resources/funding

Some threat actors are well funded with broad resources, whereas others are not. Some threat actors self-fund; others find outside investors or paying customers. Self-funded threat actors might highjack or use advertisement platforms to obtain funds; others may use ransomware to extort money from their victims. Some hackers offer their services like mercenaries to clients who pay the attackers to harm a specific target or craft a new exploit for a particular vulnerability. Some actors are paid by individuals, some by corporations, and others by nation-states.

Intent/motivation

The intent or motivation of an attacker can be unique to the individual or overlap with your own. Some attackers are motivated by the obvious benefit of money and notoriety. Others attack from boredom or just to prove to themselves that they can. Others find a thrill in the attack or are encouraged by the challenge. Some attackers are just drones in a crime group who have a daily boring drudgery of attacking a list of targets provided to them (think cubicle farms of script kiddies). Some attackers do it for fun, and others out of necessity (to earn money to feed their families). Some attack based on philosophy, political ideology, religious views, perspective on the environment, or disagreement with a business plan. Sometimes attackers have motivations that we will never know about—or might not comprehend even if we learn about them.

Use of open-source intelligence

Open-source intelligence is the gathering of information from any publicly available resource. This includes websites, social networks, discussion forums, file services, public databases, and other online sources. It also includes non-Internet sources, such as libraries and periodicals. Any information that may have been distributed by a target or by any other entity about the target is the focus of open-source intelligence gathering. The process, techniques, and methodologies used to collect open-source intelligence can be called reconnaissance, information gathering, footprinting, fingerprinting, or target research in hacking methodologies.

Exam Essentials

Define a threat actor. A threat actor is the person or entity who is responsible for causing or controlling any security-violating incidents experienced by an organization or individual.

Define script kiddies. Script kiddies are threat actors who are less knowledgeable than a professional skilled attacker. A script kiddie is usually unable to program their own attack tools and may not understand exactly how the attack operates.

Define a hacktivist. A hacktivist is someone who uses their hacking skills for a cause or purpose. A hacktivist commits criminal activities to further their cause.

Understand how organized crime is involved in cybercrime. Organized crime is involved in cybercrime activities because it is yet another area of exploitation that may allow them to gain access, power, or money.

Understand how nation-states are using cyberattacks. Most nation-states are now using cyberattacks as yet another weapon in their arsenal against their real or perceived enemies, whether internal or outside their borders.

Define APT. APT (advanced persistent threat) is any form of cyberattack that is able to continually exploit a target over a considerable period of time. An APT often takes advantage of flaws not publicly known and tries to maintain stealth throughout the attack.

Understand the risks presented by insiders. One of the biggest risks at any organization is its own internal personnel. Hackers work hard to gain what insiders already have: physical presence within the facility or a working user account on the IT infrastructure.

Understand the risks presented by competitors. While it is widely known that such actions are illegal, many organizations still elect to perform corporate espionage and sabotage against their competition.

Understand the risks presented by internal and external threat actors. Threats can originate from inside your organization as well as outside. All too often, companies focus most of their analysis and security deployment efforts on external threats without providing sufficient attention to the threats originating from inside.

Understand threat actors' level of sophistication. Threat actors can vary greatly as to their skill level and level of sophistication. Some attackers are highly trained professionals who are applying their education to malicious activities, whereas others are simply bad guys who learned how to perform cyberattacks just to expand their existing repertoire.

Know how threat actors access resources and funding. Some threat actors are well funded with broad resources; others are not. Some threat actors self-fund, whereas others find outside investors or paying customers. Self-funded threat actors might highjack or use advertisement platforms to obtain funds; others may use ransomware to extort money from their victims.

Understand threat actors' intent and motivation. The intent or motivation of an attacker can be unique to the individual or may be similar to your own. Some attackers are motivated by the obvious benefits of money and notoriety. Others attack from boredom or just to prove to themselves that they can.

Understand open-source intelligence. Open-source intelligence is the gathering of information from any publicly available resource. This includes websites, social networks, discussion forums, file services, public databases, and other online sources. It also includes non-Internet sources, such as libraries and periodicals.

1.4 Explain penetration testing concepts.

Penetration testing is a form of security evaluation that involves the same tools, techniques, and methodologies used by criminal hackers but is performed by security professionals. Penetration testing is also known as *ethical hacking* or *pen testing*.

Active reconnaissance

Active reconnaissance is collecting information about a target through interactive means. By directly interacting with a target, a person can quickly collect accurate and detailed information, but at the expense of potentially being identified as an attacker rather than just an innocent, benign, random visitor. Examples of activities that are considered active reconnaissance include visiting the target's website, performing port scanning (discussed next), speaking with the target's tech support or help desk service, visiting their physical location, and performing vulnerability scans against the target's systems.

One common function or task performed during active reconnaissance is *port scanning*. A port scanner is a vulnerability assessment tool that sends probe or test packets to a target system's ports in order to learn about the status of those ports. A port can be in one of two states: open or closed. If a valid request for connection is sent to an open TCP port (a SYN flagged packet), a normal response can be expected (a SYN/ACK flagged packet). If the TCP port is closed, the response is an RST packet. However, if a firewall is present, the firewall can filter out connection attempts on closed ports, resulting in no packet being received by the probing system. This is known as *filtering*. Thus, a TCP port scanner will have direct proof that a port is open or closed but can assume a filtered port if no response is received.

Although this form of probing works effectively, it produces traffic that is likely recorded or logged by the target system or the firewall protecting it. Thus, many other forms of port scanning have been developed. Some scanning techniques use standard packets but in an unexpected context, such as FIN or ACK flagged packets. These packets have no valid meaning outside of a valid TCP setup or teardown handshake; thus when used out of context, they may illicit a response that is meaningful to the probing entity. Even a normal data packet, which doesn't have any header flags enabled, can be used in a *NULL scan*. There are even some methods of scanning that use invalid packet constructions, such as the Xmas scan, which has numerous header flags enabled (see the earlier section "Xmas attack").

The details of how these scans operate are a bit beyond the Security+ content. However, it's important to understand that port scans allow security testers and hackers to discover what ports are open on a system. Once the open ports are known on a target, this information can lead to other important details, such as the identity of the host OS and what types of services are hosted on the target. Many port-scanning tools, such as Nmap, can not only detect open, closed, and stealth ports, but also determine the OS and identify active services on a port. Sometimes these actions are performed using a database of characteristics, and sometimes they're performed using banner-grabbing queries. A *banner grab* (see Chapter 2, "Technologies and Tools") occurs when a request for data or identity is sent to a service on an open port and that service responds with information that may directly or indirectly reveal its identity.

Passive reconnaissance

Passive reconnaissance is the activity of gathering information about a target without interacting with the target. Instead, information is collected from sources not owned and controlled by the target (other websites and services) as well as by eavesdropping on communications from the target. A significant amount of information can be gathered through passive reconnaissance, but it may not be as accurate as data gathering through active means. Additionally, eavesdropping-based reconnaissance may require a significant length of time in order to gather useful information, because you will be waiting for the transmission of the data you wish to obtain based on the normal activities of the target.

Examples of activities performed during passive reconnaissance include visiting social network sites, reading third-party reports, searching discussion forums (not operated by the target), researching domain name and IP address registrations, and visiting any other online or offline source not owned, controlled, or monitored by the target.

Pivot

In penetration testing (or hacking in general), a *pivot* is the action or ability to compromise a system, and then use the privileges or access gained through the attack to focus attention on another target that may not have been visible or exploitable initially. It is the ability to adjust the focus or the target of an intrusion after an initial foothold is gained. It is potentially possible to pivot from the compromise of computer A to launch attacks against computer B, when computer B was not accessible earlier, or once computer A is compromised, information hosted on computer A can be accessed. This could include files, database contents, security settings, and account credentials.

Pivoting also relates to *daisy chaining*, the concept of performing several exploitations in a series in order to achieve a goal on the target. Often with modern defense-in-depth or diversity-of-defense security infrastructures, a single attack is insufficient to achieve a compromise goal. Instead, several successive attacks must be waged, each dependent on and building on the success of the previous exploits. For example, a daisy chaining attack could include an initial port scan to find an open port, followed by an exploitation of the application behind the open port, which executes code to change the firewall configuration to open another port, which leads to compromising another service (which was previously inaccessible), in order to launch an injection attack, which dumps out user account credentials. This series of attack events is also an example of pivoting, because each successful attack leads to another target and exploitation.

Initial exploitation

The *initial exploitation* in a penetration test or a real-world malicious attack is the event that grants the attacker/tester access to the system. It is the first successful breach of the organization's security infrastructure that grants the attacker/tester some level of command control or remote access to the target. All steps prior to the initial exploitation—reconnaissance, port scanning, enumeration, and vulnerability detection—lead up to and make possible the initial exploitation. Once the initial exploitation is successful, the later stages of attack can occur: establishing persistent connect and control over the target and hiding all traces of the intrusion.

Persistence

Persistence is the characteristic of an attack that maintains remote access to and control over a compromised target. Some attacks are quick one-off events where the initial compromise triggers some result, such as stealing data, planting malware, destroying files, or crashing the system. But such events are short-lived "one-time, then done" occurrences, not persistent attacks. A persistent attack grants the attacker ongoing prolonged access to and control over a victim system and/or network.

Escalation of privilege

Escalation of privilege is any attack or exploit that grants the attacker greater privileges, permissions, or access than may have been achieved by the initial exploitation or that a legitimate user was assigned. Privilege escalation can be either horizontal or vertical. A horizontal privilege escalation occurs when an attack is able to jump from controlling one lower-level user account into controlling a high-level user account. This is often accomplished through credential theft using keystroke loggers planted through the initial account's capabilities, which might be installing the malware directly or sending a Trojan horse installer via email attachment. A vertical privilege escalation occurs when an attack exploits a flaw in the system or software that makes the current user account a member of an admin group, converts the account into an admin account, or simply enables the execution of commands as the system or root.

Black box

It's important to understand various terms for penetration (and other forms of) testing. A *black box* is literally a device whose internal circuits, makeup, and processing functions are unknown but whose outputs in response to various kinds of inputs can be observed and analyzed. Black-box penetration testing proceeds without using any initial knowledge of how an organization is structured, what kinds of hardware and software it uses, or its security policies, processes, and procedures.

Black-box testing requires that the penetration testers spend significant time and effort during the earlier phases of hacking to discover as much as possible about the operations of the "black box" of the target network and systems. This causes a black-box test to take the most time and cost the most (among the black, white, and gray testing options), but it provides a realistic external criminal hacker perspective on the security stance of an organization.

White box

By contrast, a *white box* is a device whose internal structure and processing are known and understood. This distinction is important in penetration testing, where white-box testing makes use of knowledge about how an organization is structured, what kinds of hardware and software it uses, and its security policies, processes, and procedures. It could be called invisible box or transparent box testing. White-box testing seeks to exploit everything known about the operations and functions of the network to focus and guide testing efforts. White-box penetration testing uses all available knowledge to drive its efforts.

White-box testers need not devote significant time and effort to reconnaissance, but perform only enough initial research activities to confirm the information provided. The overall time for a white-box test is much shorter and thus it costs significantly less as well. However, the result is that it gives a rogue administrator a lot of information about the organization's security. This is the type or form of penetration testing that is most over overlooked or discounted, because it is hard for organizational leaders to conceive of their most trusted IT administrators ever turning on them.

Gray box

Gray-box testing combines the two other approaches to perform an evaluation based on partial knowledge of the target environment. This requires some time spent on reconnaissance, and costs are usually between those of white- and black-box testing. The results are a security evaluation from the perspective of a disgruntled employee. An employee has some knowledge of the organization and its security and has some level of physical and logical access.

Pen testing vs. vulnerability scanning

Vulnerability scanning and penetration testing are important aspects of detecting and responding to new vulnerabilities and weaknesses. In addition to these important tools, ongoing monitoring of performance, throughput, and protocol use can reveal trends toward downtime, change in job focus, and the need for infrastructure upgrades.

A penetration test is a form of vulnerability scan that is performed by a special team of trained, white-hat security specialists rather than by an internal security administrator using an automated tool. Penetration testing (also known as ethical hacking) uses the same tools, techniques, and skills of real-world criminal hackers as a methodology to test the deployed security infrastructure of an organization. Penetration testing gives you the perspective of real hackers, whereas typical vulnerability scanning offers only the security perspective of the scanner's vendor.

To best simulate a real-life situation, penetration testing is usually performed without the IT or security staff being aware of it. Senior management often schedules ethical hacking events. This allows the penetration test to assess the performance of the infrastructure and the response personnel. This is known as an *unannounced test*. An *announced test* means everyone in the organization knows the penetration assessment is taking place and when.

Penetration tests can take many forms, including hacking in from the outside, simulating a disgruntled employee, social engineering attacks, and physical attacks, as well as remote connectivity, wireless, and VPN attacks. The goal of penetration testing is to discover weaknesses before real criminals do. Most penetration testing requires high levels of knowledge and skill on the part of the testers. Automated tools are employed, but most of the benefit derived from a penetration test is from the skill of the testers modifying existing exploits or crafting custom code for attacks. This is because real hackers often write their own surgically precise attack tools and scripts based on their target. Security administrators do use automated tools for vulnerability scanning to check for policy compliance and known issues. Penetration testing is used to discover new weaknesses that these automated tools can't find.

In security terms, a *penetration* occurs when an attack is successful and an intruder is able to breach the perimeter around your environment. A breach can be as small as reading a few bits of data from your network or as big as logging in as a user with unrestricted privileges. A primary goal of security is to prevent penetrations.

One common method you can employ to test the strength of your security measures is to perform penetration testing, a vigorous attempt to break into your protected network using any means available. It's common for organizations to hire external consultants to perform penetration testing so testers aren't privy to confidential elements of the environment's security configuration, network design, and other internal secrets.

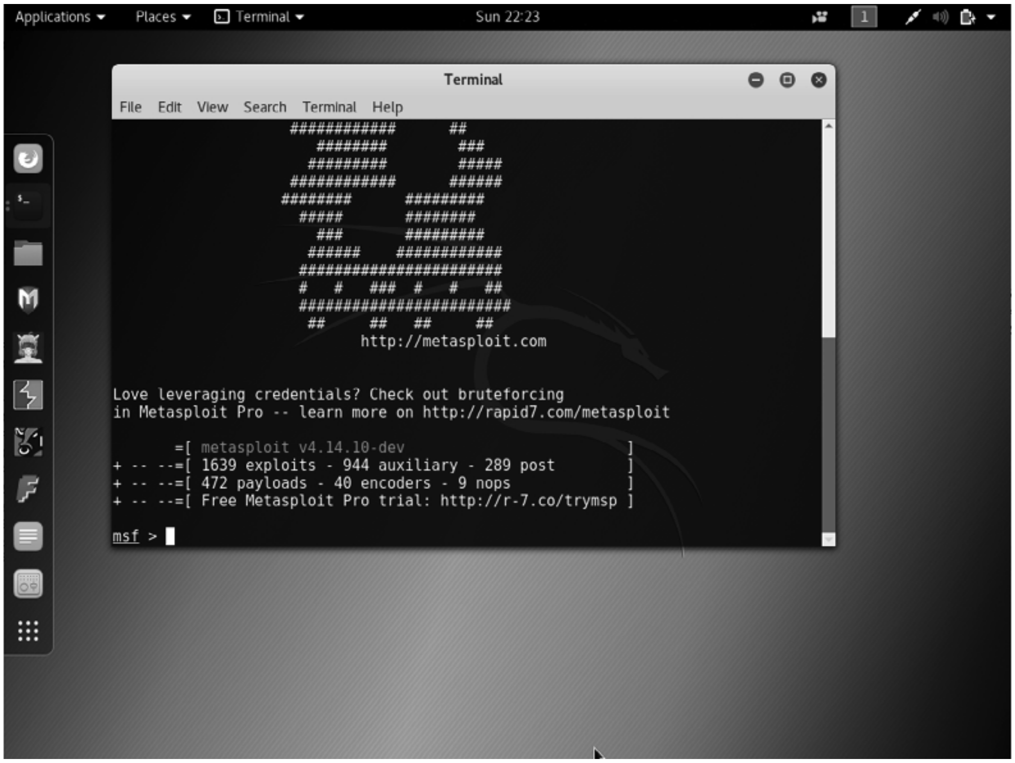
Penetration testing seeks to find any and all detectable weaknesses in your existing security perimeter. The operative term is *detectable*; there are undetected and presently unknowable threats lurking in the large-scale infrastructure of network software and hardware design that no amount of penetration testing can directly discover. Once a weakness is discovered, countermeasures can be selected and deployed to improve security in the environment. One significant difference between penetration testing and an actual attack is that once a vulnerability is discovered during a penetration test, the intrusion attempt ceases before a vulnerability exploit can cause any damage. There are open-source and commercial tools (such as Metasploit [Figure 1.14], Immunity's CANVAS, and CORE Impact) that can be considered active security scanners or exploitation frameworks; they allow you to take penetration testing one step further and attempt to exploit known vulnerabilities in systems and networks. These tools may be used by good guys and bad guys alike.

Penetration testing may use automated attack tools or suites or may be performed manually using common network utilities and scripts. Automated attack tools range from professional vulnerability scanners to wild, underground tools discovered on the Internet. Tools are also often used for penetration testing that's performed manually, but the real emphasis is on knowing how to perpetrate an attack.

Penetration testing should be performed only with the consent and knowledge of management (and security staff). Performing unapproved security testing could cause productivity losses, trigger emergency response teams, or even cost you your job and potentially earn you jail time.

Regularly staged penetration tests are a good way to accurately judge the security mechanisms deployed by an organization. Penetration testing can also reveal areas where patches or security settings are insufficient and where new vulnerabilities have developed. To evaluate your system, benchmarking and testing tools are available for download at `www.cisecurity.org`, and a somewhat comprehensive list of security assessment and hacker/penetration testing tools is available from `www.sectools.org`.

FIGURE 1.14 The CLI (command-line interface) of Metasploit on Kali Linux



```
                    ############      ##
                       ########        ###
                      #########        #####
                    ############      ######
                   ########      #########
                     #####       ########
                       ###       #########
                      ######    ############
                     #######################
                     #   #   ###  #   #   ##
                     ########################
                      ##     ##   ##     ##
                            http://metasploit.com


Love leveraging credentials? Check out bruteforcing
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.14.10-dev                         ]
+ -- --=[ 1639 exploits - 944 auxiliary - 289 post        ]
+ -- --=[ 472 payloads - 40 encoders - 9 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > 
```

Identifying and repelling attacks requires an explicit, well-defined body of knowledge about their nature and occurrence. Some attack patterns leave behind signatures that make them readily apparent to casual observation with IDS instrumentation; other forms of attack are esoteric or not conducive to pattern-matching engines and therefore must be measured against a baseline of acceptable activity.

What elements or properties signify an attack sequence rather than a benign traffic formation? Answering this question depends on careful, attentive security professionals keeping up with the latest attacks, vulnerabilities, exploits, and security bulletins (like those from the U.S. Computer Emergency Readiness Team at `www.us-cert.gov/cas/bulletins` or those from the Common Vulnerabilities and Exposures database at `http://cve.mitre.org`).

Before implementing a fix or a security control, it's important to verify that a problem actually exists. There is no point in protecting against a threat if your environment doesn't have the vulnerability. Likewise, if the threat doesn't exist or is extremely unlikely to ever become realized in your organization, implementing countermeasures may also be unwarranted.

Part of penetration testing is to confirm whether a vulnerability exists and whether a real threat exists. Based on the criticality of known threats, vulnerabilities, and risks, you can determine whether to respond by implementing a countermeasure, assigning the risk elsewhere, or accepting the risk.

Hackers often attempt to find a way to bypass security controls. An ethical hacker or penetration tester attempts many of these same techniques so that you can be aware of them before they're abused by someone malicious. Means of bypassing security controls vary greatly, but some common general categories include using alternate physical or logical pathways, overloading controls, and exploiting new flaws. If hackers know that a specific pathway of approach is secured, they may seek an alternate route. For example, if all Internet-sourced traffic is filtered by a firewall, a hacker may try to locate a modem or an unauthorized wireless access point on the network to bypass the firewall's security.

Sometimes DoS/DDoS attacks can be used to overload firewalls, IDS, IPS, auditing, and so on, so that these security tools are "distracted" while the real attack takes place. Also, new exploits are being crafted daily that may be able to compromise security through exploitation of faulty programming code. For examples, see the Exploit Database at `www.exploit-db.com` for a current list of exposed new and zero-day exploits.

Just because an electronic lock or other form of access control is in use, that doesn't ensure that bypassing the system is impossible. Ways to bypass electronic controls include turning off the power, creating a short circuit, introducing an alternative power supply, bypassing triggering circuits, and overloading detectors with false positives.

A penetration test should be used to find new flaws or unknown vulnerabilities as well as to test the abilities of the deployed security infrastructure. If current security controls aren't sufficient or can be easily bypassed, a thorough penetration test should reveal this. If your security posture isn't resilient enough to catch proficient ethical hackers, then it's unlikely that it's good enough to catch professional criminal hackers.

A penetration test should discover vulnerabilities and then exploit them to a predetermined extent. The testing should not be performed to the point of causing unrepairable damage or prolonged downtime. The whole point of penetration testing is for the testers to act ethically and within restrictions or boundaries imposed by the service-level agreement (SLA) or testing contract. Any test that might cause harm should gain specific preapproval before it's executed. Additionally, the target being tested should be prepared with recent backups and a recovery team just in case the tester's precautions aren't sufficient or the attack accidentally is more extensive than expected.

Exam Essentials

Understand active reconnaissance. Active reconnaissance is the idea of collecting information about a target through interactive means. By interacting with a target, accurate and detailed information can be collected quickly but at the expense of potentially being identified as an attacker rather than just an innocent, benign, random visitor.

Know how to use port scanners. A port scanner is a vulnerability assessment tool that sends probe or test packets to a target system's ports in order to learn about the status of those ports.

Understand passive reconnaissance. Passive reconnaissance is the activity of gathering information about a target without interacting with the target. Instead, information is collected from sources not owned and controlled by the target (other websites and services) as well as by eavesdropping on communications from the target.

Define pivoting. In penetration testing (or hacking in general), a pivot is the action or ability to compromise a system, and then using the privileges or access gained through the attack to focus attention on another target that may not have been visible or exploitable initially.

Understand initial exploitation. The initial exploitation in a penetration test or a real-world malicious attack is the event that grants the attacker/tester access to the system. It is the first successful breach of the organization's security infrastructure that grants the attacker/tester some level of command control or remote access to the target.

Define persistence. Persistence is the concept of an attack that maintains remote access to and control over a compromised target. A persistent attack grants the attacker ongoing prolonged access to and control over a victim system and/or network.

Understand escalation of privilege. Escalation of privilege is any attack or exploit that grants the attacker greater privileges, permissions, or access than what may have been achieved by the initial exploitation. Privilege escalation can be either horizontal or vertical.

Understand black-box testing. Black-box penetration testing proceeds without using any initial knowledge of how an organization is structured; what kinds of hardware and software it uses; or its security policies, processes, and procedures. It provides a realistic external criminal hacker perspective on the security stance of an organization.

Understand white-box testing. White-box testing makes use of knowledge about how an organization is structured, what kinds of hardware and software it uses, and its security policies, processes, and procedures. The result is that it gives a rogue administrator a lot of information about the organization's security.

Understand gray-box testing. Gray-box testing combines the two other approaches to perform an evaluation based on partial knowledge of the target environment. The results are a security evaluation from the perspective of a disgruntled employee.

Understand penetration testing. A penetration test is a form of vulnerability scan that is performed by a special team of trained white-hat security specialists rather than by an internal security administrator using an automated tool. Penetration testing (also known as ethical hacking) uses the same tools, techniques, and skills of real-world criminal hackers as a methodology to test the deployed security infrastructure of an organization.

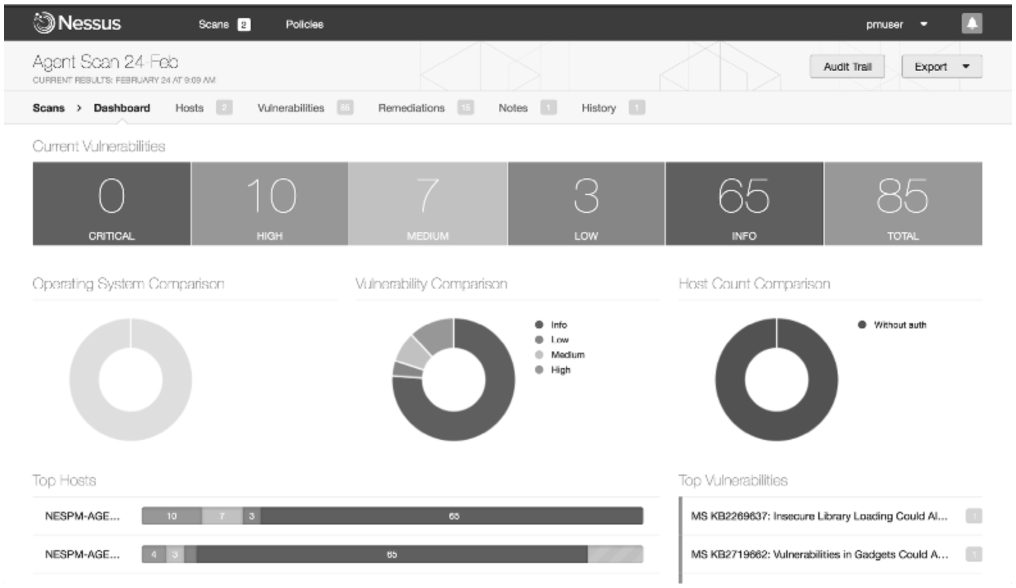
1.5 Explain vulnerability scanning concepts.

Vulnerability scanning is used to discover weaknesses in deployed security systems in order to improve or repair them before a breach occurs. By using a wide variety of assessment tools (such as vulnerability scanners, protocol analyzers, network scanners, and wireless scanners), security administrators can learn about deficiencies quickly. Only through vigilance and constant monitoring and assessment can a security endeavor prove successful.

Typically, vulnerability scanning should be performed by security administrators on a regular periodic basis (such as weekly). Additionally, only after thoroughly performing vulnerability scanning and responding to/addressing each alert item is an organization ready for a true penetration test. A penetration test requires dedicated full-time testing professionals, who are often external consultants. A vulnerability scan can be run by any reasonably skilled and knowledgeable IT or security administrator with a little training and lab testing.

Vulnerability scanners and security-assessment tools are used to test a system for known security vulnerabilities and weaknesses. They're used to generate reports that indicate the aspects of the system that need to be managed to improve security. The reports may recommend applying patches or making specific configuration or security setting changes to improve or impose security (Figure 1.15).

FIGURE 1.15 The scan summary report from the vulnerability scanner Nessus



A vulnerability scanner is only as useful as its database of security issues. Thus, the database must be updated from the vendor often to provide a useful audit of your system. The use of vulnerability scanners in conjunction with an IDS may help reduce false positives by the IDS and keep the total number of overall intrusions or security violations to a minimum. When discovered vulnerabilities are patched quickly and often, the system provides a more secure environment.

An extension to the concept of the IDS is the IPS. An IPS seeks to actively block unauthorized connection attempts or illicit traffic patterns as they occur. IPS designs fall under the same type (host- and network-based) and classification (behavior- and signature-based) as the IDS counterparts, and they're often deployed together for complete network coverage. Additionally, many IPS platforms are capable of dissecting higher-level application protocols in search of malicious payloads. The line between IDSs and IPSs can be blurred in that many self-professed IDSs have IPS capabilities. These days, detection and prevention systems occur together more often than they do separately.

The results of a vulnerability scan need to be interpreted by a knowledgeable security expert. Automated scanning tools can produce numerous false positives; thus it may be necessary to confirm the presence of a security flaw before implementing a fix, especially if the fix is costly or interferes with production. Another issue is that the criticality level reported by a scanning tool may not be accurate or relevant to your organization. Finally, the results of a vulnerability scan must be interpreted in light of the existing environment, known real threats, and budget.

Passively test security controls

A passive test of security controls is being performed when an automated vulnerability scanner is being used that seeks to identify weaknesses without fully exploiting discovered vulnerabilities. In most cases, automated vulnerability scanners detect the security control as it attempts a test. Additionally, because the security controls are operating while the automated vulnerability scan is being performed, the security controls get a workout at the same time the actual targets are the focus of the scan. Thus, passively testing security controls takes place any time tests are performed against targets but not specifically directed toward the security measures themselves.

Actively testing security controls involves attempting to fully exploit and breach a target system. This might be performed using active scanners, also known as exploitation frameworks, or using manual attacks.

Identify vulnerability

A scanner that is able to identify a vulnerability does so through a testing probing process defined in its database of evaluations. The goal of a vulnerability scanner is to inform you of any potential weaknesses or attack points on your network, within a system, or against an individual application. In most cases, a vulnerability scanner evaluates a target using surface probing activities, which does not fully exploit the potential flaw. Thus the report is listing all issues based on the symptoms of a vulnerability, not the confirmed result of an actual exploitation. This causes some of the items on the report to be false positives. Thus, it is important for system managers to investigate each item on a vulnerability scanner's report to confirm whether or not they are actual exploitable flaws before undertaking any mitigation activities.

Identify lack of security controls

An important task for a vulnerability scanner is to identify any necessary or best-practice security controls that are not present in the evaluated target. Such a report may indicate that updates and patches are not applied or that a specific security mechanism is not present, such as encryption, antivirus scanning, a firewall, and so on. If a vulnerability scanner can easily determine that your environment is missing key elements of a security infrastructure, so, too, can a hacker discover this and take advantage of your lack of sufficient protection.

Identify common misconfigurations

Many vulnerability scanners can determine whether or not you have improper, poor, or misconfigured systems and protections. If a vulnerability scanner is able to detect this issue, so can an attacker. Be sure to correct any discovered misconfigurations immediately.

Intrusive vs. non-intrusive

An *intrusive* vulnerability scan (also known as *active evaluation*) attempts to exploit any flaws or vulnerabilities detected. A *nonintrusive* vulnerability scan (also known as *passive evaluation*) only discovers the symptoms of flaws and vulnerabilities and doesn't attempt to exploit them. Traditionally, a vulnerability scanner is assumed to be nonintrusive, whereas a penetration test is assumed to be intrusive. However, a range of assessment tools can now provide either form of evaluation.

Credentialed vs. non-credentialed

A *credentialed* scan is one where the logon credentials of a user, typically a domain administrator, must be provided to the scanner in order for it to perform its work. The account credentials provided are most likely a domain account rather than a local account, such as root or administrator. A *noncredentialed* scan is one where no user accounts are provided to the scanning tool, so only those vulnerabilities that don't require credentials are discovered. Both forms of scanning should be used to provide a thorough evaluation of your security infrastructure.

False positive

A *false positive* is the occurrence of an alarm or alert due to a benign activity being initially classified as potentially malicious. The problem with false positives is they cause security administrators to waste time investigating nonmalicious events. Over time, and after repeated false positives, security admins may stop responding to alarms and assume all alerts are false.

An even more important issue to address is the *false negative*. Whereas a false positive is an alarm without a malicious event, a false negative is a malicious event without an alarm. When false negatives occur, it is assumed that only benign events are occurring; however, malicious activities are actually taking place. This is the equivalent of a building burning without fire alarms.

A *false negative* occurs when an alarm or alert is not triggered by malicious or abnormal events. False negatives occur when poor detection technologies are used, when detection databases are not kept current, or when an organization is facing a new, unknown zero-day threat. When malicious activities are occurring and are not detected, the victim is unaware of the situation. They are actively being harmed while not being aware that the harm is occurring. Thus, they do not know that they need to make any response or adjustment. This is the realm of the unknown unknown.

	Malicious events	**Benign events**
Alarm/alert	True positive	False positive
No alarm/alert	False negative	True negative

To reduce the risk of false negatives, organizations should adopt a deny-by-default or implicit-deny security stance. This stance centers on the idea that nothing is allowed to occur, such as execution, unless it is specifically allowed (placed on a whitelist or an exception list). It is also good practice to keep detection technologies, such as firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs), current in terms of their core engines as well as their rule lists and detection databases.

Exam Essentials

Understand vulnerability scanning. Vulnerability scanning is used to discover weaknesses in deployed security systems in order to improve or repair them before a breach occurs. By using a wide variety of assessment tools, security administrators can learn about deficiencies quickly.

Understand passive testing of security controls. A passive test of security controls is being performed when an automated vulnerability scanner is being used that seeks to identify weaknesses without fully exploiting discovered vulnerabilities.

Understand vulnerability identification. A scanner that is able to identify a vulnerability does so through a testing probing process defined in its database of evaluations. The goal of a vulnerability scanner is to inform you of any potential weaknesses or attack points on your network, within a system, or against an individual application.

Understand the identification of a lack of security controls. An important task for a vulnerability scanner is to identify any necessary or best-practice security controls that are not present in the evaluated target. Such a report may indicate that updates and patches are not applied or that a specific security mechanism is not present.

Be able to identify common misconfigurations. Many vulnerability scanners can determine whether or not you have improper, poor, or misconfigured systems and protections. If a vulnerability scanner is able to detect this issue, so can an attacker.

Understand intrusive vs. nonintrusive. An intrusive vulnerability scan attempts to exploit any flaws or vulnerabilities detected (also known as active evaluation). A nonintrusive vulnerability scan only discovers the symptoms of flaws and vulnerabilities and doesn't attempt to exploit them (also known as passive evaluation).

Understand credentialed vs. noncredentialed. A credentialed scan is one where the logon credentials of a user, typically a system administrator or the root, must be provided to the scanner in order for it to perform its work. A noncredentialed scan is one where no user accounts are provided to the scanning tool, so only those vulnerabilities that don't require credentials are discovered.

Know what a false positive is. A false positive occurs when an alarm or alert is triggered by benign or normal events.

Know what a false negative is. A false negative occurs when an alarm or alert is not triggered by malicious or abnormal events.

1.6 Explain the impact associated with types of vulnerabilities.

There are many different forms and types of hacks, attacks, exploits, and intrusions. Many of these compromises can cause significant harm or damage to a system as well as impede the ability of an organization to continue normal operations. This section focuses on the impact that some forms of vulnerabilities and their exploitation may have on an organization.

Race conditions

Computer systems perform tasks with rigid precision. Computers excel at repeatable tasks. Attackers can develop attacks based on the predictability of task execution. The common sequence of events for an algorithm is to check that a resource is available and then access it if you are permitted. The time of check (TOC) is the time at which the subject checks on the status of the object. There may be several decisions to make before returning to the object to access it. When the decision is made to access the object, the procedure accesses it at the time of use (TOU). The difference between the TOC and the TOU is sometimes large enough for an attacker to replace the original object with another object that suits their own needs. *Time-of-check-to-time-of-use (TOCTTOU)* attacks are often called *race conditions* because the attacker is racing with the legitimate process to replace the object before it is used.

A classic example of a TOCTTOU attack is replacing a data file after its identity has been verified but before data is read. By replacing one authentic data file with another file of the attacker's choosing and design, an attacker can potentially direct the actions of a program in many ways. Of course, the attacker would have to have in-depth knowledge of the program and system under attack.

Likewise, attackers can attempt to take action between two known states when the state of a resource or the entire system changes. Communication disconnects also provide small windows that an attacker might seek to exploit. Anytime a status check of a resource precedes action on the resource, a window of opportunity exists for a potential attack in the brief interval between check and action. These attacks must be addressed in your security policy and in your security model. TOCTTOU attacks, race condition exploits, and communication disconnects are known as state attacks because they attack timing, dataflow control, and transition between one system state and another.

Another form of race condition attack occurs when two processes are running concurrently but one is designed to finish first and then provide its results to the second process in order for it to complete its tasks. If the first process is delayed in completing its task, this may cause the second process to be vulnerable to injection of malicious content (since it is not receiving the needed input from the first process), or it may cause the second process to fail.

Race condition attacks can result in system takeover, data leakage, and data destruction.

Vulnerabilities due to:

Every nontypical and specialized system places unique and often complex security strains on your organization. It is important to keep these in mind when designing your security policy and performing network segmentation.

End-of-life systems

End-of-life systems are those that are no longer receiving updates and support from the vendor. If an organization continues to use an end-of-life system, then the risk of compromise is high because any future exploitation will never be patched or fixed. It is of utmost important to move off end-of-life systems in order to maintain a secure environment. It might not seem initially cost-effective or practical to move away from a solution that still works, just because the vendor has terminated support. However, the security management efforts you will expend will likely far exceed the cost of developing and deploying a modern system–based replacement.

Embedded systems

An *embedded system* is any form of computing component added to an existing mechanical or electrical system for the purpose of providing automation and/or monitoring. Embedded systems can be a security risk because they are generally static systems, meaning that even the administrators who deploy them have no real means to alter the device's operations in order to address security vulnerabilities. Some embedded systems can be updated with patches from the vendor, but often patches are released months after a known exploit is found in the wild. It is essential that embedded systems be isolated from the Internet and from a private production network in order to minimize exposure to remote exploitation, remote control, or malware compromise.

Lack of vendor support

Any system, whether hardware or software, will become more insecure over time once it lacks vendor support. Lack of support can be a "feature" of the product all along, where the vendor does not provide any improvement, support, or patching/upgrading of the product after the initial sale. As a security manager, you should avoid products that lack vendor support and phase out products as they reach their end-of-life date.

Improper input handling

Many forms of exploitation are caused by the lack of *input sanitization* or validation. Only with proper input handling can software exploitation be reduced or eliminated. There are three main forms of input filtering that should be adopted by every programmer and included in every code they author:

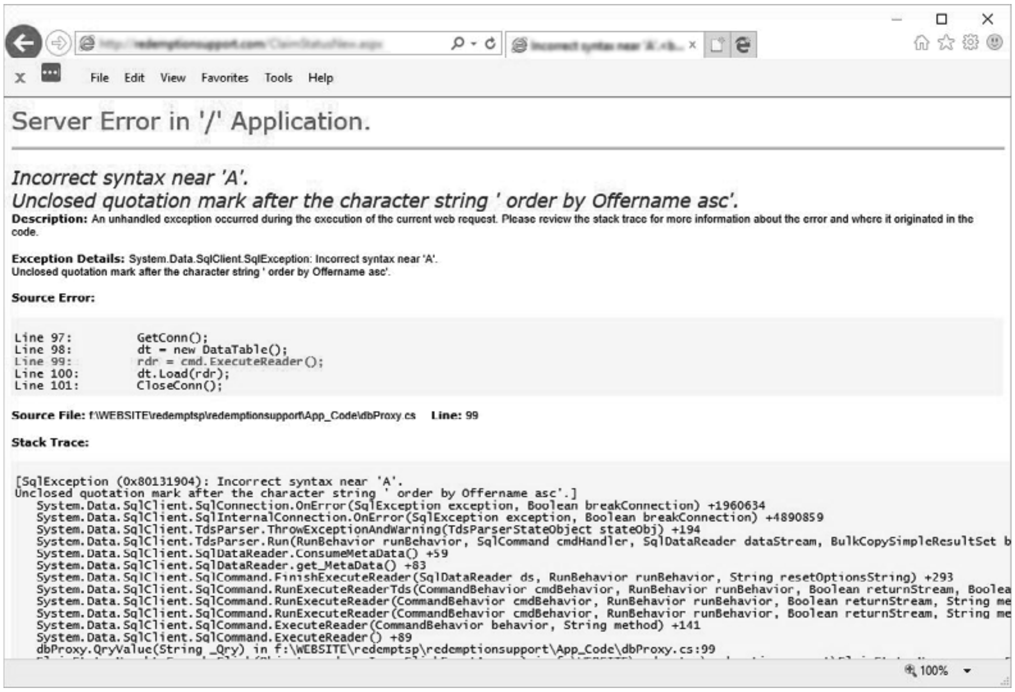
- Check for length.
- Filter for known malware patterns.
- Escape metacharacters.

Improper error handling

Improper error handling may allow for the leaking of essential information to attackers or enable attackers to force a system into an insecure state. If error messages are not handled properly, they may disclose details about a flaw or weakness that will enable an attacker to fine-tune their exploit. For example, if an attacker submits just a single quote to a target system, if the error response indicates that there is an unclosed quotation mark (Figure 1.16), then it informs the attacker that no metacharacter filtering is taking place. Otherwise, the error would have stated that invalid or out-of-bounds input was attempted but was rejected and that the user should try again.

If errors themselves are not handled properly, it could cause an application to disclose confidential data to a visitor, allow an attacker to bypass authentication, or even crash the system. Programmers should include an error management system in their products in order to handle invalid values, out-of-range data sets, or other forms of improper input. When an error is detected, the error management system should display a generic error message to the user, such as "error try again" or "error, contact technical support." The error management system should log all details about the error into a file for the administrator, but should not disclose those details to the user. Additionally, if an error could result in security violations, a general error fault response known as fail-secure should be initiated. A fail-secure system will revert to a secured, closed, protective state in the event of a failure rather than into an open, insecure, nonprotected state where information can be disclosed or modified.

FIGURE 1.16 An error page for a website that shows the lack of metacharacter filtering



Server Error in '/' Application.

Incorrect syntax near 'A'.
Unclosed quotation mark after the character string ' order by Offername asc'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Incorrect syntax near 'A'.
Unclosed quotation mark after the character string ' order by Offername asc'.

Source Error:

```
Line 97:        GetConn();
Line 98:        dt = new DataTable();
Line 99:        rdr = cmd.ExecuteReader();
Line 100:       dt.Load(rdr);
Line 101:       CloseConn();
```

Source File: f:\WEBSITE\redemptsp\redemptionsupport\App_Code\dbProxy.cs **Line:** 99

Stack Trace:

```
[SqlException (0x80131904): Incorrect syntax near 'A'.
Unclosed quotation mark after the character string ' order by Offername asc'.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection) +1960634
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection) +4890859
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj) +194
   System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet b
   System.Data.SqlClient.SqlDataReader.ConsumeMetaData() +59
   System.Data.SqlClient.SqlDataReader.get_MetaData() +83
   System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString) +293
   System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolea
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String me
   System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String me
   System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +141
   System.Data.SqlClient.SqlCommand.ExecuteReader() +89
   dbProxy.QryValue(String _Qry) in f:\WEBSITE\redemptsp\redemptionsupport\App_Code\dbProxy.cs:99
```

Misconfiguration/weak configuration

It is the responsibility of system implementers and those performing ongoing system management to verify that the correct and secure configuration items remain defined and enforced. When *misconfigurations* or *weak configurations* are allowed to remain while a system is in active productive use, the risk of data loss, data leakage, and overall system compromise is higher.

Default configuration

Default configurations should never be allowed to remain on a device or within an application. Defaults are intended for ease of installation and initial configuration in order to minimize support calls from new customers. As a system administrator, you should alter system settings from their defaults to a state that brings the system into compliance with your security policy. The tyranny of the default is the fact that defaults are usually insecure and thus leave a system open to simple compromise.

Resource exhaustion

Resource exhaustion occurs when applications are allowed to operate in an unrestricted and unmonitored manner so that all available system resources are consumed in the attempt to serve the requests of valid users or in response to a DoS attack. It is essential for system managers to monitor the baseline of productive valid resource consumption and watch for trends that may indicate a need to expand capacity or to respond to exploitative attacks.

Untrained users

Untrained users are more likely to make mistakes or abuse a system's resources and capabilities. Only trained workers should be allowed to use sensitive system resources. Organizations need to train new employees properly on test systems not directly tied to production. Only once a new employee shows proficiency in accomplishing tasks should he or she be moved into a live production system.

Improperly configured accounts

User accounts need to be properly configured in order to grant the correct level of resource access and system rights based on the job responsibilities of the employees. No matter what the means of authorization, users should be granted only enough powers to accomplish their work tasks. Any more than that minimum is simply increasing risk for the organization without any benefit. Improperly configured accounts violate the principle of least privilege. Workers should have only the object permissions or privileges and system user rights or capabilities that they need for their specific jobs.

Vulnerable business processes

All business tasks, processes, procedures, or functions should be assessed as to their importance to the organization and their relative vulnerabilities. This includes considering the confidentiality, integrity, and availability protections or deficiencies for each business process. Attention should also be given to the value and importance of the data sets that each business process processes.

Weak cipher suites and implementations

Not all ciphers or other algorithm elements in a cipher suite are secure. Many older algorithms or implementations of algorithms have known flaws, weaknesses, or means of compromise. These weaker ciphers should be avoided and disabled and replaced with stronger cipher suites with few or no issues. A cipher's age isn't necessarily an indication of strength or weakness. For a discussion about weak ciphers, cipher suite attacks, and Google's recommendations for the future, read "A roster of TLS cipher suites weaknesses" at `http://googleonlinesecurity.blogspot.com/2013/11/a-roster-of-tls-cipher-suites-weaknesses.html`.

Memory/buffer vulnerability

Memory is a key element of a computer system. It is the area holding data that was received as input, whether from the keyboard, network, or storage device. The area of memory set aside or assigned to hold input is known as a buffer. Memory or buffer attacks and exploits are serious security concerns.

Memory leak

A *memory leak* is the opposite of what the name might imply. A memory leak occurs when a program fails to release memory or continues to consume more memory. It's called a leak because the overall computer system ends up with less available free memory when an application is causing a memory leak. It might be more appropriate to call this issue a memory consumption flaw. Depending on the speed of the memory leak, the issue may not be noticeable in typical circumstances (such as when an application is closed after a few minutes of use) or may quickly degenerate, causing system failures. Programmers should focus on properly managing memory and releasing memory allocations once they are no longer needed. Otherwise, end users and system administrators should monitor system performance for software memory leaks and then elect to discontinue the use of offending products.

Integer overflow

An *integer overflow* is the state that occurs when a mathematical operation attempts to create a numeric value that is too large to be contained or represented by the allocated storage space or memory structure. For example, an 8-bit value can only hold the numbers 0 to 255. If an additional number is added to the maximum value, an integer overflow occurs. Often, the number value resets or rolls over to 0, similar to the way a vehicle odometer rolls over. However, in other cases, the result *saturates*, meaning the maximum value is retained. Thus, the result is another form of error (missing or lost information). In yet other cases, the rollover results in a negative number. If the programming logic assumes that a number will always be positive, then when a negative number is processed, it could have security-breaching results. Programmers need to understand the numeric limitations of their code and the platform for which they're developing. There are coding techniques programmers should adopt in order to test for integer-overflow results before an overflow can occur.

Buffer overflow

A *buffer overflow* is a memory exploitation that takes advantage of a software's lack of input length validation. By injecting larger than expected input into a system, this attack may result in the extra data "overflowing" the assigned buffer and thus overwrite memory in the following adjacent locations. Such a buffer overflow might simply cause a system freeze or execution malfunction. However, in some cases a buffer overflow can allow for the injection of shellcode (precompiled malicious code) into memory, where it may be executed with system-level privileges. This is known as a buffer overflow attack leading to arbitrary code execution. The primary defense against buffer overflow is input sanitization, specifically limiting the length of input.

Pointer dereference

*A pointer dereferenc*e is the programmatic activity of retrieving the value stored in a memory location by triggering the pulling of the memory based on its address or location as stored in a pointer (a type of variable that holds an address—that is, a memory space location). Invalid dereferencing can occur due to attempting to dereference a pointer that was not initialized (assigned a memory address), dereferencing a pointer that retrieves data to be assigned to a variable that is not configured as the same data type (binary vs. ASCII or numbers vs. text), and dereferencing a pointer that was deallocated due to a dynamic memory allocation change. If a programmer leaves in code that causes an invalid dereference, it could cause a crash of the application, cause the system to freeze, or even open vulnerabilities that can be exploited by other means (such as buffer overflow attacks).

DLL injection

DLL injection is an advanced software exploitation technique that manipulates a process's memory in order to trick it into loading additional code and thus performing operations the original author did not intend. A *DLL (dynamic link library)* is a collection of code that is designed to be loaded and used as needed by a process. Many DLLs are designed to perform common functions and thus are shared among many applications.

A DLL injection attack starts off by manipulating the memory of a live process in order to inject commands that trick the process into loading and executing the malicious DLL. This is similar to when you have a shopping list posted on the fridge that you grab as you head to the grocery store, only to discover when you return home that your neighbor broke into your apartment and added beer, chips, and dip to the list, and now that you have returned home with those items, he has declared that you are now hosting a poker party for him and his questionable friends.

System sprawl/undocumented assets

System sprawl or *server sprawl* is the situation where numerous underutilized servers are operating in your organization's server room. These servers are taking up space, consuming electricity, and placing demands on other resources, but their provided workload or productivity does not justify their presence. This can occur if an organization purchases cheap lower-end hardware in bulk instead of selecting optimal equipment for specific use cases. Consolidation of software onto optimized hardware designed to manage resource consumption with little resource contention is a response to system sprawl. It is also likely that using virtualization to run several guest OSs on a single hardware server can reduce the inefficiencies as well.

Undocumented assets are another form of wasted resources and lost opportunity. Without clear knowledge of what equipment is present in an organization, it is impossible to plan for future growth, adopt proper security measures, or track down offending elements. Every asset used in a business task should be identified and tracked. This will help maximize the production potential of existing hardware while minimizing the purchase of unneeded, superfluous, or ill-suited equipment.

Architecture/design weaknesses

Architecture or design flaws are distinct from coding bugs. Bugs are mistakes in the authoring of the software code, often typographical errors or the use of the wrong function. *Design flaws* are mistakes in the overall concept, theory, implementation, or structure of an application. Design flaws may exist because of a misunderstanding of the problem that was intended to be solved, not understanding the requirements of the solution, violating common or good practice design principles, or failing to account for security measures during initial conception.

Architecture or design flaws often fall into three main categories: omission flaws, commission flaws, or realization flaws. Omission flaws occur when a security requirement is overlooked or a key element of the development process is ignored. Commission flaws occur when poor decisions were made about how to perform certain actions, resolve problems, or strike a balance between performance and security. Realization flaws occur when the correct design concept was selected but it was improperly implemented in code, thus causing a problem that is indistinguishable from an omission or commission flaw.

For a more thorough explanation of design flaws, see Common Architecture Weakness Enumeration (CAWE) at `http://blog.ieeesoftware.org/2016/04/common-architecture-weakness.html` and visit the CWE (common weakness enumeration) catalog hosted by MITRE at `http://cwe.mitre.org/`.

New threats/zero day

New threats are being developed by hackers on a nearly daily basis. It is an essential part of security management to be aware of new threats. Performing daily research can assist you in remaining up to date. To see or track some of the concerns, security professionals can review various websites for threat information. Some useful sites of this ilk are `https://www.exploit-db.com`, `https://cve.mitre.org`, `https://nvd.nist.gov/`, and `https://www.us-cert.gov`.

By keeping an eye on the security trends and alerts related to new zero-day compromises, you will be better prepared to respond to incidents as well as defend against them.

Thousands of new virus and malware variations are crafted and released daily. Fortunately, only a small portion of these are significant threats. However, that is not cause to overlook the severity of the damage that even a single malicious code infection could cause.

Everyone needs a current antivirus scanner. This scanner should be configured to download updates daily on an automatic schedule. The system should be scanned fully at least once per week. The system's activity should be monitored in real time. Although antivirus software has advanced significantly in the last few years, it is still not a substitute for avoiding risky activity and controlling user behavior.

Please see the earlier coverage of zero-day issues in the sections "Application/Service Attacks" and "Zero Day."

Improper certificate and key management

Key management is always a concern when cryptography is involved. Most of the failures of a cryptosystem are based on improper key management rather than on the algorithms. Good key selection is based on the quality and availability of random numbers. Most mobile devices must rely locally on poor random number–producing mechanisms or access more robust random number generators (RNGs) over a wireless link. Once keys are created, they need to be stored in such a way as to minimize exposure to loss or compromise. The best option for key storage is usually removable hardware or the use of a trusted platform module (TPM), but these are rarely available on mobile phones and tablets.

For more discussion on key management in general, see the section "Key Escrow" in Chapter 6, "Cryptography and PKI."

Exam Essentials

Understand race conditions. Time-of-check-to-time-of-use (TOCTTOU) attacks are often called race conditions because the attacker is racing with the legitimate process to replace the object before it is used. Another form of race condition attack occurs when two processes are running concurrently and one process is designed to finish first, but the attack alters the processing to change the order of completion.

Comprehend end-of-life systems. End-of-life systems are those that are no longer receiving updates and support from their vendors. If an organization continues to use an end-of-life system, then the risk of compromise is high because no future exploitation will ever be patched or fixed.

Understand embedded systems. An embedded system is any form of computing component added to an existing mechanical or electrical system for the purpose of providing automation and/or monitoring.

Realize that there may be a lack of vendor support. Any system, whether hardware or software, will become more insecure over time once it lacks vendor support. The lack of vendor support can be due to end-of-life dropping of support, but it can also be a "feature" of the product all along, where the vendor does not provide any improvement, support, or patching/upgrading of the product after the initial sale.

Understand improper input handling. Many forms of exploitation are caused by the lack of input sanitization or validation. Only with proper input handling can software exploitation be reduced or eliminated.

Know proper input handling. There are three main forms of input filtering that should be adopted by every programmer and included in every code they author: check for length, filter for known malware patterns, and escape metacharacters.

Understand improper error handling. Improper error handling may allow for the leaking of essential information to attackers or enable attackers to force a system into an insecure state. If error messages are not handled properly, they may disclose details about a flaw or weakness that will enable an attacker to fine-tune their exploit.

Understand misconfiguration/weak configuration. When misconfigurations or weak configurations are allowed to remain while a system is in active productive use, the risk of data loss, data leakage, and overall system compromise is higher.

Know the risks of default configuration. Default configurations should never be allowed to remain on a device or within an application. The tyranny of the default is the fact that defaults are usually insecure and thus leave a system open to simple compromise.

Understand resource exhaustion. Resource exhaustion occurs when applications are allowed to operate in an unrestricted and unmonitored manner so that all available system resources are consumed in the attempt to serve the requests of valid users or in response to a DoS attack.

Understand untrained users. Untrained users are more likely to make mistakes or abuse a system's resources and capabilities.

Understand improperly configured accounts. The concept of improperly configured accounts is a violation of the principle of least privilege.

Understand vulnerable business processes. All business tasks, processes, procedures, and functions should be assessed as to their importance to the organization and their relative vulnerabilities.

Understand weak cipher suites and implementations. Many older algorithms or implementations of algorithms have known flaws, weaknesses, or means of compromise. These weaker ciphers should be avoided and disabled and replaced with stronger cipher suites with few or no issues.

Understand memory leaks. A memory leak occurs when a program fails to release memory or continues to consume more memory.

Understand integer overflow. An integer overflow is the state that occurs when a mathematical operation attempts to create a numeric value that is too large to be contained or represented by the allocated storage space or memory structure.

Understand buffer overflow. A buffer overflow is a memory exploitation that takes advantage of a software's lack of input length validation. In some cases a buffer overflow can allow for the injection of shellcode (precompiled malicious code) into memory, where it may become executed with system-level privileges.

Understand pointer dereference. Pointer dereferencing is the programmatic activity of retrieving the value stored in a memory location by triggering the pulling of the memory based on its address or location as stored in a pointer.

Understand DLL injection. DLL injection is an advanced software exploitation technique that manipulates a process's memory in order to trick it into loading additional code and thus perform operations the original author did not intend.

Comprehend system sprawl/undocumented assets. System sprawl or server sprawl is the situation where numerous underutilized servers are operating in your organization's server room. The existence of undocumented assets is a form of wasted resources and lost opportunity.

Understand architecture/design weaknesses. Architecture or design flaws are mistakes in the overall concept, theory, implementation, or structure of an application. Design flaws may exist because of a misunderstanding of the problem that was intended to be solved, not understanding the requirements of the solution, violating common or good practice design principles, or failing to account for security measures during initial conception.

Understand new threats. New threats are being developed by hackers on a nearly daily basis. It is an essential part of security management to be aware of new threats.

Understand improper certificate and key management. Most of the failures of a cryptosystem are based on improper key management rather than on the algorithms.

Review Questions

You can find the answers in the Appendix.

1. An attacker has decided to attempt to compromise your organization's network. They have already determined the ISP you are using and know your public IP addresses. They have also performed port scanning to discover your open ports. What communications technique can the hacker now use to identify the applications that are running on each open port facing the Internet?
   - **A.** Credentialed penetration test
   - **B.** Intrusive vulnerability scan
   - **C.** Banner grabbing
   - **D.** Port scanning
2. You are the security manager for a large organization. Your NIDS has reported abnormal levels of network activity and several systems have become unresponsive. While investigating the causes of these issues, you discover a rootkit on your mission-critical database server. What is the best step to take to return this system to production?
   - **A.** Reconstitute the system.
   - **B.** Run an antivirus tool.
   - **C.** Install a HIDS.
   - **D.** Apply vendor patches.
3. If user awareness is overlooked, what attack is more likely to succeed?
   - **A.** Man-in-the-middle
   - **B.** Reverse hash matching
   - **C.** Physical intrusion
   - **D.** Social engineering
4. A pirated movie-sharing service is discovered operating on company equipment. Administrators do not know who planted the service or who the users are. What technique could be used to attempt to trace the identity of the users?
   - **A.** Typo squatting
   - **B.** Integer overflow
   - **C.** Watering hole attack
   - **D.** Ransomware
5. You are the IT security manager for a retail merchant organization that is just going online with an e-commerce website. You hired several programmers to craft the code that is the backbone of your new web sales system. However, you are concerned that while the new code functions well, it might not be secure. You begin to review the code, systems design, and services architecture to track down issues and concerns. Which of the following do you hope to find in order to prevent or protect against XSS?
   - **A.** Input validation
   - **B.** Defensive coding

C. Allowing script input

D. Escaping metacharacters

6. What type of virus attempts to disable security features that are focused on preventing malware infection?

A. Retrovirus

B. Polymorphic

C. Companion

D. Armored

7. What does the acronym RAT stand for?

A. Random Access Token

B. Remote Authentication Testing

C. Random Authorization Trajectory

D. Remote Access Trojan

8. What form of social engineering attack focuses on stealing credentials or identity information from any potential target?

A. Phishing

B. Tailgating

C. Dumpster diving

D. Logic bomb

9. What type of service attack positions the attacker in the communication path between a client and a server?

A. Session hijacking

B. Man-in-the-middle

C. Amplification

D. Replay

10. What form of attack abuses a program's lack of length limitation on the data it receives before storing the input in memory and can lead to arbitrary code execution?

A. ARP poisoning

B. XSS

C. Domain hijacking

D. Buffer overflow

11. What is a programmatic activity that restricts or reorganizes software code without changing its externally perceived behavior or produced results?

A. Buffer overflow

B. Pass the hash

C. Refactoring

D. Shimming

12. What wireless attack is able to trick mobile device users into connecting into its man-in-the-middle style of attack by automatically appearing as if it is a trusted network that they have connected to in the past?
    - A. Replay
    - B. Evil twin
    - C. Bluesnarfing
    - D. Disassociation
13. What type of hacker hacks for a cause or purpose, knowing that they may be identified, apprehended, and prosecuted?
    - A. Hacktivist
    - B. Script kiddie
    - C. Nation-state hacker
    - D. Internal attacker
14. When an attacker selects a target, they must perform reconnaissance to learn as much as possible about the systems and their configuration before launching attacks. What is the term for the gathering of information from any publicly available resource, such as websites, social networks, discussion forums, file services, and public databases?
    - A. Banner grabbing
    - B. Port scanning
    - C. Open-source intelligence
    - D. Enumeration
15. What penetration testing or hacking term refers to the concept of continuing an intrusion after an initial compromise in order to further breach an organization by focusing on new targets that may not have been accessible initially?
    - A. Man-in-the-browser
    - B. Pivot
    - C. Daisy chaining
    - D. Shimming
16. What is the term for an attack or exploit that grants the attacker greater privileges, permissions, or access than what may have been achieved by the initial exploitation?
    - A. Hoax
    - B. Impersonation
    - C. Piggybacking
    - D. Privilege escalation

17. What type of information-gathering tactics rely on direct interaction with the target while attempting to avoid being detected as malicious?

A. Passive reconnaissance

B. Banner grabbing

C. Active reconnaissance

D. Social engineering

18. What type of test of security controls is performed with an automated vulnerability scanner that seeks to identify weaknesses while listening in on network communications?

A. Active

B. Passive

C. External

D. Noncredentialed

19. What is the term used to describe systems that are no longer receiving updates and support from their vendors?

A. Passive

B. Embedded

C. End-of-life

D. Static

20. What is present on a system for ease of installation and initial configuration in order to minimize support calls from new customers?

A. Default configuration

B. Resource exhaustion trigger

C. Buffer overflow flaw

D. Collision tool