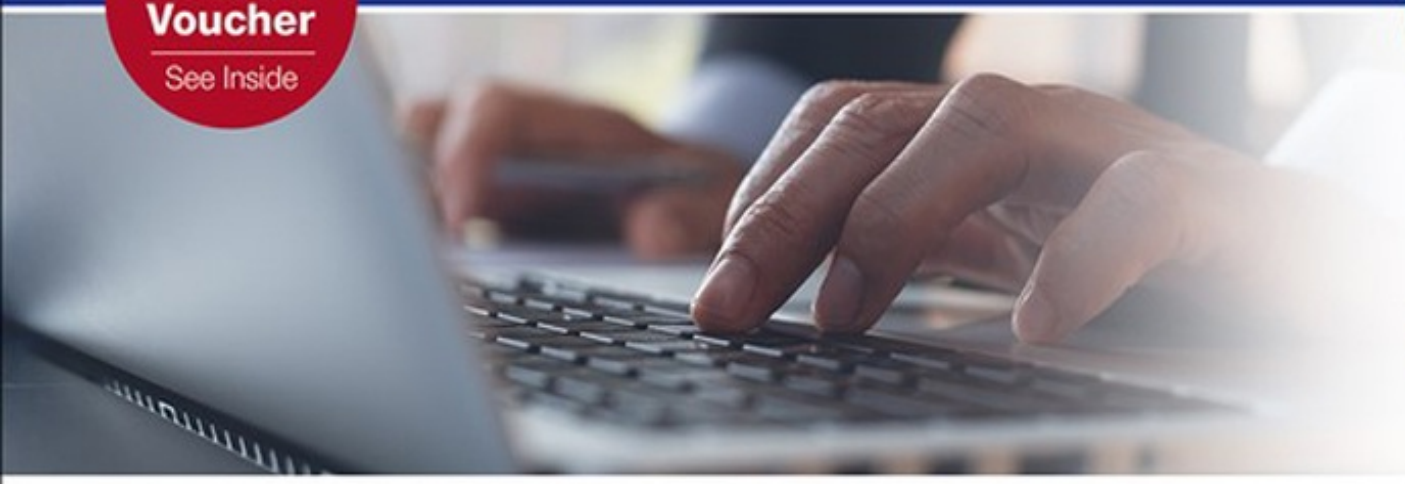
Save 10% on Exam Voucher

See Inside



Practice Tests

Flash Cards

Review Exercises

Study Planner

Cert Guide

Advance your IT career with hands-on learning

CompTIA®

Security+

SY0-601

OMAR SANTOS
RON TAYLOR
JOSEPH MLODZIANOWSKI

PEARSON IT CERTIFICATION

CompTIA Security+ SY0-601 Cert Guide

Omar Santos
Ron Taylor
Joseph Mlodzianowski

Pearson IT Certification

Contents

Table of Contents

Part I: Threats, Attacks, and Vulnerabilities

Chapter 1. Comparing and Contrasting Different Types of Social Engineering Techniques

This chapter covers the following topics related to Objective 1.1 (Compare and contrast different types of social engineering techniques.) of the CompTIA Security+ SY0-601 certification exam:

- Phishing
- Smishing
- Vishing
- Spam
- Spam over internet messaging (SPIM)
- Spear phishing
- Dumpster diving
- Shoulder surfing
- Pharming
- Tailgating
- Eliciting information
- Whaling
- Prepending
- Identity fraud
- Invoice scams
- Credential harvesting
- Reconnaissance

- Hoax
- Impersonation
- Watering hole attack
- Typo squatting
- Pretexting
- Influence campaigns (hybrid warfare and social media)
- Principles of social engineering and reasons for effectiveness (including Authority, Intimidation, Consensus, Scarcity, Familiarity, Trust, and Urgency)

Social engineering attacks leverage the weakest link, which is typically the human user. If an attacker can get a user to reveal information, it is much easier for the attacker to cause harm than it is by using some other method of reconnaissance. Social engineering can be accomplished through email or misdirection of web pages, prompting a user to click something that leads to the attacker gaining information. Social engineering can also be done in person by an insider or an outside entity or over the phone.

This chapter delves into the methods and techniques that social engineers can employ to gain access to buildings and systems and obtain company data and personal information. It also covers the various ways that these social engineers can be defeated.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 1-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 1-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Social Engineering Fundamentals	1–9
User Security Awareness Education	10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. An attacker sends a targeted email with a malicious attachment to a user in your company. This attacker researched public information about the user to send a "more personal" and targeted email to the user. Which of the following is this type of attack?

a. Spear phishing

b. Typo squatting

c. Pharming

d. None of these answers are correct.

2. Which of the following is an example of a tool that can be used specifically to perform social engineering attacks?

a. Maltego

b. SET

c. The Harvester

d. Recon-NG

3. Which of the following best describes the difference between smishing and vishing?

a. Vishing is social engineering attack in which the attacker calls the user over the phone and then persuades the user to reveal sensitive information or perform a given action. Smishing is a type of phishing campaign using SMS text messages instead of email.

b. Vishing is social engineering attack in which the attacker leaves a voicemail and then persuades the user to reveal sensitive information or perform a given action. Smishing is a type of typo squatting and pharming campaign using Bluetooth.

c. Vishing is social engineering attack in which the attacker leaves a voicemail and then persuades the user to reveal sensitive information or perform a given action. Smishing is a type of typo squatting and pharming campaign using short Internet messaging systems.

d. None of these answers are correct.

4. An _____ is a small space that can usually only fit one person, used to combat tailgating.

a. tunnel-gap

b. tunnel-trap

c. piggyback

d. access control vestibule

5. Which of the following best describes what is pretexting?

a. Impersonation

b. Social engineering

c. Whaling

d. Pharming

6. Which of the following refers to the act of incorporating malicious ads on trusted websites, which results in users' browsers being inadvertently redirected to sites hosting malware?

a. Malvertising

b. Pharming

c. Active ad exploitation

d. Whaling

7. Which of the following is true about spear phishing?

a. Spear phishing attacks use the Windows Administrative Center.

b. Spear phishing is phishing attempts that are constructed in a very specific way and directly targeted to specific individuals or companies.

c. Spear phishing, whaling, and phishing are the same type of attack.

d. Spear phishing attacks use the Windows PowerShell.

8. Derek is the CEO of a Fortune 500 company. He received an email with a malicious attachment. Once Derek clicked on the attachment, malware was installed on his system. Which of the following best describes this attack?

a. Smishing

b. Vishing

c. Whaling

d. Pretexting

9. Which of the following is true about social engineering motivation techniques?

a. Social proof can be used to create a feeling of urgency in a decision-making context. It is possible to use specific language in an interaction to present a sense of urgency and manipulate the victim.

b. Scarcity can be used to create a feeling of urgency in a decision-making context. It is possible to use specific language in an interaction to present a sense of urgency and manipulate the victim.

c. Scarcity cannot be used to create a feeling of urgency in a decision-making context. It is possible to use specific language in an interaction to present a sense of urgency and manipulate your victim.

d. Social proof cannot be used in an interrogation because it is illegal. It is not legal to use specific language in an interaction to present a sense of urgency and manipulate your victim.

10. Which of the following recommendations can be used in user security awareness education?

a. Adhere to the organization's clean desk policy, which states that all documents, electronics, personally owned devices, and other items be put away (or locked away) when the user is not at his or her desk, or other work area.

b. Always screen your email and phone calls carefully and keep a log of events. This is also known as communications vetting.

c. Use encryption when possible to protect emails, phone calls, and data.

d. All of these answers are correct.

Foundation Topics

Social Engineering Fundamentals

Let's discuss a low note in our society. Because that is what information security–based social engineering is—a low form of behavior, but an effective one. It is estimated that 1 out of 10 people is conned every year through social engineering methods, and as many as half of them don't even know it has occurred. It's glorified in the movies, but in real life it can have devastating consequences to an organization and to innocent individuals.

Social engineering is the act of manipulating users into revealing confidential information or performing other actions detrimental to the user. Examples of social engineering are common in everyday life. A basic example would be a person asking for your username and password over the phone; often the person uses flattery to gain information. Malicious people use various forms of social engineering in an attempt to steal whatever you have of value: your money, information, identity, confidential company data, or IT equipment.

A primary example is attackers leveraging normal user behavior. Suppose that you are a security professional who is in charge of the network firewalls and other security infrastructure equipment in your company. An attacker could post a job offer for a very lucrative position and make it very attractive to you, the victim. Suppose the job description lists benefits and compensation far beyond what you are already making at your company. You

decide to apply for the position. The criminal (attacker) then schedules an interview with you. Because you are likely to "show off" your skills and work, the attacker may be able to get you to explain how you have configured the firewalls and other network infrastructure devices for your company. You might disclose information about the firewalls used in your network, how you have configured them, how they were designed, and so on. Your answers give the attacker a lot of knowledge about the organization without requiring the attacker to perform any type of scanning or reconnaissance on the network.

Let's take a look at another example: suppose that you are a security guard and a pregnant woman comes to you saying that she is feeling sick and that she needs a bathroom immediately. You are courteous and escort her to the bathroom inside your premises, where she then uses her laptop to connect to your Wi-Fi or put a wireless rogue access point (AP) to lure some of your users to connect to such an AP. We are good and kind people (most of the time) and will continue to be the weakest link in the cybersecurity world because this human nature becomes our weakness.

Social engineers will rely on information. For example, open-source intelligence (OSINT) is a way that attackers can gain knowledge about a target. OSINT includes media, public government data, commercial data, and academic publications. You will learn additional details about OSINT in Chapter 5, "Understanding Different Threat Actors, Vectors, and Intelligence Sources."

Tip

You can find several references to tools that can be used for OSINT reconnaissance at my GitHub repository: https://github.com/The-Art-of-Hacking/h4cker. This GitHub repository has thousands of cybersecurity references that can be very helpful for this certification and many others.

The main reason that social engineering succeeds is a lack of user awareness. But social engineering can also be effective in environments in which the IT personnel have little training, and in public areas—for example, public

buildings with shared office space. Let’s discuss some of the more common types of social engineering.

Common social engineering techniques include the following:

- Phishing and spear phishing
- Smishing
- Vishing
- Spam
- Spam over Internet messaging (SPIM)
- Dumpster diving
- Shoulder surfing
- Pharming
- Tailgating
- Eliciting information
- Whaling
- Prepending
- Identity fraud
- Invoice scams
- Credential harvesting
- Reconnaissance
- Hoax
- Impersonation or pretexting
- Eavesdropping
- Baiting
- Watering hole attack
- Typo squatting

Phishing and Spear Phishing

Phishing is the attempt at fraudulently obtaining private information. A phisher usually masquerades as someone else, perhaps another entity. There are two main differences between phishing and pretexting. ***Pretexting*** is the act of impersonating or "spoofing" someone else's identity. For example, an attacker could impersonate an employee of a company or a business partner to attempt to steal sensitive data from the victim. Phishing is usually done by electronic communication, not in person. Second, little information about the target is necessary. A phisher might target thousands of individuals without much concern as to their background. An example of phishing would be an email that requests verification of private information. The email probably leads to a malicious website designed to lure people into a false sense of security to fraudulently obtain information. The website often looks like a legitimate website. A common phishing technique is to pose as a vendor (such as an online retailer or domain registrar) and send the target email confirmations of orders that they supposedly placed.

This is a triple-whammy. First, the orders are obviously fake; a person might say, "Hey, wait! I didn't place these orders!" and perhaps click the link(s) in the email, leading the person to the false web page. Second, if a person thinks it's a legitimate order (perhaps the person does many orders, and the fraudulent one looks like another legitimate one), the person might click a link to track the order, again leading to the bogus web page. Third, once at the web page, the person is asked to enter credentials for an account (which then leads to credit card fraud and ID theft), and in addition to that the page might have Trojans and other malicious scripts that are delivered to the unsuspecting person on exit. Sheesh, talk about cyberbullying!

Generally, no information about the target is necessary for a phishing attack. However, some "phishermen" actually target specific groups of people or even specific individuals. This is known as ***spear phishing***. And when an attacker targets key stakeholders and senior executives (CEOs, CFOs, and so on), it is known as ***whaling***. Whaling attacks are much more detailed and require that the attacker know a good deal of information about the target

(much of which is freely available on the Internet).

Many different types of social engineering are often lumped into what is referred to as phishing, but actual phishing for private information is normally limited to email and websites.

Several open-source tools can help automate or accelerate a social engineering attack (including sending phishing and spear phishing emails). An example of these tools (and one of the most popular) is the Social Engineering Toolkit (SET). Example 1-1 demonstrates how attackers can use SET to send spear phishing emails to a victim and either select existing payloads or create their own.

Example 1-1 The Social Engineering Toolkit

```
[---]        The Social-Engineer Toolkit (SET)         [---]
[---]        Created by: David Kennedy (ReL1K)         [---]
                         Version: 8.0.3
                      Codename: 'Maverick'
[---]        Follow us on Twitter: @TrustedSec         [---]
[---]        Follow me on Twitter: @HackingDave        [---]
[---]       Homepage: https://www.trustedsec.com       [---]
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.
   The Social-Engineer Toolkit is a product of TrustedSec.
           Visit: https://www.trustedsec.com
   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

 Select from the menu:
   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) Third Party Modules
```

```
  99) Return back to the main menu.
set> 1

 The Spearphishing module allows you to specially craft email
messages and
 send  them to a large (or small) number of people with attached
fileformat
 malicious  payloads. If you want to spoof your email address, be
sure
 "Sendmail" is in-  stalled (apt-get install sendmail) and change
the
 config/set_config SENDMAIL=OFF  flag to SENDMAIL=ON.
 There are two options, one is getting your feet wet and letting
SET do
 everything for you (option 1), the second is to create your own
 FileFormat
 payload and use it in your own attack. Either way, good luck and
enjoy!

   1) Perform a Mass Email Attack
   2) Create a FileFormat Payload
   3) Create a Social-Engineering Template
  99) Return to Main Menu
set:phishing>1
/usr/share/metasploit-framework/
 Select the file format exploit you want.
 The default is the PDF embedded EXE.
           ********** PAYLOADS **********
   1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
   2) SET Custom Written Document UNC LM SMB Capture Attack
   3) MS15-100 Microsoft Windows Media Center MCL Vulnerability
   4) MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
   5) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
   6) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-
087)
   7) Adobe Flash Player "Button" Remote Code Execution
   8) Adobe CoolType SING Table "uniqueName" Overflow
   9) Adobe Flash Player "newfunction" Invalid Pointer Use
  10) Adobe Collab.collectEmailInfo Buffer Overflow
```

```
  11) Adobe Collab.getIcon Buffer Overflow
  12) Adobe JBIG2Decode Memory Corruption Exploit
  13) Adobe PDF Embedded EXE Social Engineering
  14) Adobe util.printf() Buffer Overflow
  15) Custom EXE to VBA (sent via RAR) (RAR required)
  16) Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
  17) Adobe PDF Embedded EXE Social Engineering (NOJS)
  18) Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
  19) Apple QuickTime PICT PnSize Buffer Overflow
  20) Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
  21) Adobe Reader u3D Memory Corruption Vulnerability
  22) MSCOMCTL ActiveX Buffer Overflow (ms12-027)
set:payloads>
```

Note

SET and other penetration testing tools are not covered in the exam. The instance provided in Example 1-1 is for your reference only. SET and similar tools can be installed in different Linux distributions. Some of the most popular Linux distributions for security penetration testing are Kali Linux, Parrot Security, and Black Arch.

To defend against this, users should install a phishing filter or add-on and enable it on the web browser. Also, a person should be trained to realize that institutions will *not* call or email requesting private information. If people are not sure, they should hang up the phone or simply delete the email. A quick way to find out whether an email is phishing for information is to hover over a link. You will see a URL domain name that is far different from that of the institution that the phisher is claiming to be, probably a URL located in a distant country. Many of these phishers are also probably engaging in spy-phishing: a combination of spyware and phishing that effectively makes use of spyware applications. A spyware application of this sort is downloaded to the target, which then enables additional phishing attempts that go beyond the initial phishing website.

Smishing

Because phishing has been an effective tactic for threat actors, they have found ways other than using email to fool their victims into following malicious links or activating malware from emails. A number of phishing campaigns have used Short Message Service (SMS) to send malware or malicious links to mobile devices. This social engineering technique is also known as ***smishing*** (short for SMS phishing).

One example of SMS phishing is the Bitcoin-related SMS scams that have surfaced in recent years. Numerous victims have received messages instructing them to click links to confirm their accounts and claim Bitcoins. When a user clicks such a link, he or she might be fooled into entering sensitive information on that attacker's site.

Vishing

Phishing attacks can also be accomplished by telephone conversations. Phone or voice phishing, known as ***vishing***, works in the same manner as phishing but is initiated by a phone call (often using VoIP systems). Vishing is typically used to steal sensitive information such as Social Security numbers, credit card numbers, or other information used in identity theft schemes. Attackers might impersonate and spoof caller ID to obfuscate themselves when performing voice phishing attacks.

In many vishing attacks, the phone call often sounds like a prerecorded message from a legitimate institution (bank, online retailer, donation collector, and so on). The message asks the unsuspecting person for confidential information such as name, bank account numbers, codes, and so on—all under the guise of needing to verify information for the person's protection. It's really the opposite, of course, and many people are caught unawares by these types of scams every day. Through the use of automated

systems (such as the ones telemarketers use), vishing can be perpetuated on large groups of people with little effort.

Note

A similar technique using automated systems is known as *war-dialing*. This type of attack occurs when a device (modem or other system) is used to scan a list of telephone numbers and dial them in search of computer systems and fax machines. The technique sifts out the phone numbers associated with voice lines and the numbers associated with computers. The result is a list that can later be used by other attackers for various purposes.

Spam and Spam over Internet Messaging (SPIM)

Spam has traditionally been the term used to describe unwanted and unsolicited email messages. However, email messages are not the only way that attackers or malicious users send spam. Users may also receive spam (numerous messages) via social media sites and instant/Internet messaging applications. For instance, you can be talking to other security professionals over a Discord server and someone may hop on the server and send dozens of unwanted messages, sometimes protesting against something. A better example is receiving unwanted messages and scam offers via your Facebook Messenger, WhatsApp, Keybase, Signal, WeChat, Telegram, or any other instant messaging application.

Tip

Several free services enable you to obtain DNS-based blocking lists of spammers, as well as ways that you can report spam events. An example is SpamCop (https://www.spamcop.net).

Dumpster Diving

When a person literally scavenges for private information in garbage and recycling containers, that is known as ***dumpster diving***. Any sensitive documents should be stored in a safe place as long as possible. When they are no longer necessary, they should be shredded. (Some organizations incinerate their documents.) Information might be found not only on paper but also on hard drives or removable media. Chapter 15, "Understanding the Importance of Physical Security Controls," covers proper recycling and/or destruction of hard drives in more detail.

Shoulder Surfing

Shoulder surfing occurs when a person uses direct observation to find out a target's password, PIN, or other such authentication information. The simple resolution for this type of behavior is for the user to shield the screen with privacy screens or filters, keypads, or other authentication-requesting devices. A more aggressive approach is to courteously ask the suspected shoulder surfer to move along. Also, private information should never be left on a desk or out in the open. Computers should be locked or logged off when the user is not in the immediate area. From a more technical perspective, password masking can be implemented (if not already), where typed passwords appear only as asterisks or dots on the screen. Always check if your systems, applications, and devices use password masking. Some lesser devices (such as SOHO routers) may not implement password masking by default, and that might go against company policy due to the inherent lack of security. Shoulder surfing, along with eavesdropping, and dumpster diving are examples of no-tech hacking.

Pharming

Pharming is the term used to describe a threat actor redirecting a victim from a valid website or resource to a malicious one that could be made to appear as the valid site to the user. From there, an attempt is made to extract confidential information from the user or to install malware in the victim's system. Pharming can be done by altering the hosts file on a victim's system, through DNS poisoning, or by exploiting a vulnerability in a DNS server.

Figure 1-1 illustrates the mechanics of how pharming works.

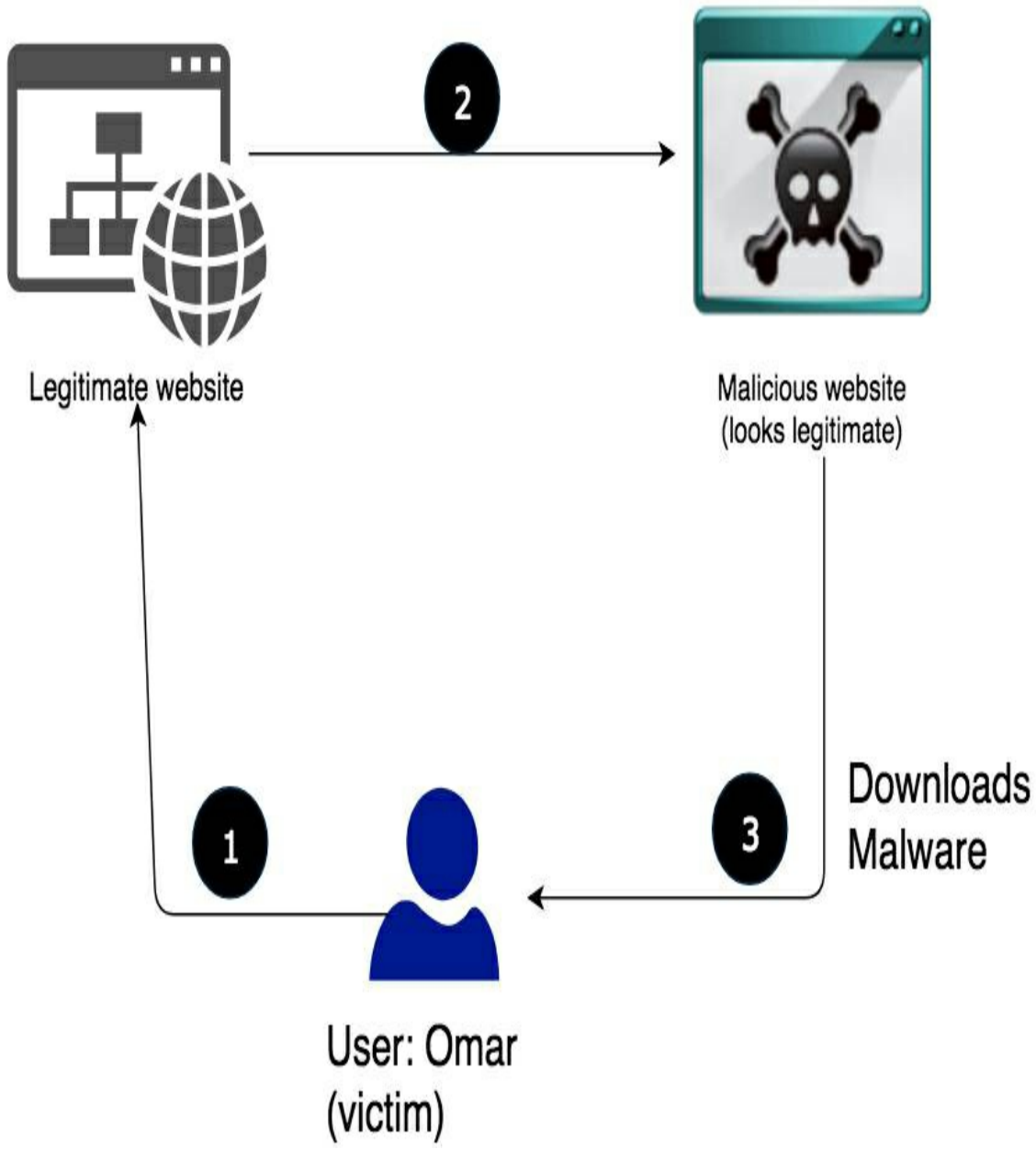


Figure 1-1 Pharming Example

The following steps are illustrated in Figure 1-1:

1. The user (Omar) visits a legitimate website and clicks a legitimate link.

2. Omar's system is compromised, the hosts file is modified, and Omar is redirected to a malicious site that appears to be legitimate. (This could also be accomplished by compromising a DNS server or spoofing a DNS reply.)

3. Malware is downloaded and installed on Omar's system.

Piggybacking or Tailgating

When an unauthorized person tags along with an authorized person to gain entry to a restricted area—usually with the person's consent—that person is ***piggybacking***. ***Tailgating*** is essentially the same with one difference: it is usually without the authorized person's consent. Both of these attacks can be defeated through the use of access control vestibules (formerly known as mantraps). An access control vestibule is a small space that can usually fit only one person. It has two sets of interlocking doors; the first set must be closed before the other will open, creating a sort of waiting room where people are identified (and cannot escape!). This technique is often used in server rooms and data centers. Multifactor authentication is often used in conjunction with an access control vestibule; for example, using a proximity card and PIN at the first door and using a biometric scan at the second. An access control vestibule is an example of a preventive security control. Turnstiles, double entry doors, and security guards are other less expensive (and less effective) solutions to the problem of piggybacking and tailgating and help address confidentiality in general.

Eliciting Information

Key components of social engineering are how someone influences, interrogates, and impersonates others. In short, ***eliciting information*** is the act of gaining knowledge or information from people. In most cases, an attacker gets information from the victim without directly asking for that particular information.

How an attacker interrogates and interacts with a victim is crucial for the success of the social engineering campaign. An interrogator can ask good open-ended questions to learn about an individual's viewpoints, values, and

goals. The interrogator can then use any information revealed to continue to gather additional information or to obtain information from another victim.

It is also possible for an interrogator to use closed-ended questions to get more control of the conversation and to lead the conversation or to stop the conversation. Asking too many questions can cause the victim to shut down the interaction, and asking too few questions might seem awkward. Successful social engineering interrogators use a narrowing approach in their questioning to gain the most information from the victim.

Interrogators pay close attention to the following:

- The victim's posture or body language
- The coloring of the victim's skin, such as the victim's face color becoming pale or red
- The direction of the victim's head and eyes
- Movement of the victim's hands and feet
- The victim's mouth and lip expressions
- The pitch and rate of the victim's voice, as well as changes in the voice
- The victim's words, including their length, the number of syllables, dysfunctions, and pauses

With pretexting—or impersonation—an attacker presents as someone else in order to gain access to information. In some cases, this type of social engineering attack can be very simple, such as quickly pretending to be someone else within an organization; in other cases, it can involve creating a whole new identity and then using that identity to manipulate the receipt of information. Social engineers might use pretexting to impersonate individuals in certain jobs and roles, even if they do not have experience in those jobs or roles.

For example, a social engineer might impersonate an IT support worker and provide unsolicited help to a user. Impersonating IT staff can be very effective because if you ask someone if he or she has a technical problem, it is quite likely that the victim will think about it and say something like, "Yes, as a matter of fact…yesterday, this weird thing happened to my computer." Impersonating IT staff can give an attacker physical access to systems in the

organization. The attacker who has physical access can use a USB stick containing custom scripts to compromise a computer within seconds.

Whaling

Whaling is similar to phishing and spear phishing; however, with whaling, the attack is targeted at high-profile business executives and key individuals in a corporation. So, what is the difference between whaling and spear phishing? Like threat actors conducting spear phishing attacks, threat actors conducting whaling attacks also create emails and web pages to serve malware or collect sensitive information; however, the whaling attackers' emails and pages have a more official or serious look and feel. Whaling emails are designed to look like critical business emails or something from someone who has legitimate authority, either externally or even internally in the company itself. In whaling attacks, web pages are designed to specifically address high-profile victims. In a regular phishing attack, the email might be a faked warning from a bank or service provider. In whaling attacks, the email or a web page would be created with a more serious executive-level form. The content is created to target an upper manager, such as the CEO, or an individual who might have credentials for valuable accounts within the organization. In summary, a whaling attack takes additional steps to target and entice higher-profile victims.

The main goal in whaling attacks is to steal sensitive information or compromise the victim's system and then target other key high-profile victims.

Attackers could use multifaceted attacks (also known as combined social engineering attacks). For instance, an attacker could send a spear-phishing email to a victim and then follow up with a phone call. This makes the attack even more effective.

Prepending

You can often configure your email servers or email cloud services to prepend a message in the email subject line to identify emails that are coming from outside of the organization. This technique is also known as ***prepending***. For example, the subject line of an email could read something like this:

```
Subject: "[EXTERNAL] Order Update"
```

Identity Fraud

Social engineering has been used by many threat actors and scammers to steal sensitive information to then open new bank accounts, make purchases, get tax refunds, among many other scenarios. The United States Federal Trade Commission (FTC) created an educational website to help you report and recover from identity theft at https://www.identitytheft.gov. Other countries have similar websites and resources.

As stated by the United States FTC, “There are many ways that you might discover that someone is using your information. You might get a notice from the IRS or find unfamiliar accounts on your credit report. You might notice strange withdrawals from your bank account, get bills that aren’t yours, or get calls about debts that you don’t owe.”

Invoice Scams

Many scammers continue to send unsolicited emails and other messages to victims that include “an invoice” (as a malicious attachment) for something that they have not purchased. For instance, a scammer might send you a targeted email message including an invoice for something that you most

likely have purchased in the past. The scammer may look for information about you (using OSINT) and then send a targeted email with such invoice. For example, the attacker/scammer may have found your LinkedIn profile and then will send an email with a fake invoice from LinkedIn. Because you use that service, you might be more susceptible to click on that attachment and, subsequently, for your system to get compromised.

Credential Harvesting

Nowadays, you hear about a new breach pretty much on a daily basis. In many cases, attackers dump user credentials in sites like Pastebin, GitHub, and many others. Consequently, it is easier for attackers to compromise systems because users reuse their usernames and passwords to authenticate to other sites and systems. ***Credential harvesting*** (often called account harvesting) refers to the attacking technique or activities of grabbing legitimate usernames and even passwords to gain access to systems to steal information or to use them for malicious purposes.

Reconnaissance

Threat actors obtain many details about their victims before launching an attack. They can do this by actively performing scans of systems and networks (active reconnaissance). However, threat actors can also leverage open-source intelligence (OSINT) or human intelligence (HUMINT) to potentially obtain sensitive information from a target without the need of using active scans or sending any packets to the targeted network and underlying systems (this is referred to as passive reconnaissance).

Tip

The MITRE Corporation (MITRE) is a United States

government-funded research organization. It has created different standards and frameworks that are used by many organizations and cybersecurity professionals. MITRE created a set of matrices (called the MITRE ATT&CK) that document the tactics and techniques attackers use to compromise networks and systems. As part of the same effort, MITRE maintains a list of tactics and techniques used by attackers to gather information about their victims before compromising their systems and networks (PRE-ATT&CK). MITRE's ATT&CK can be accessed at https://attack.mitre.org and PRE-ATT&CK can be accessed at https://attack.mitre.org/tactics/pre

---

Hoaxes

A ***hoax*** is the attempt at deceiving people into believing something that is false. The differences between hoaxes and phishing can be quite gray. However, hoaxes can come in person or through other means of communication, whereas phishing is generally relegated to e-communication and phone. Although phishing can occur at any time, and with the specific goal of obtaining private information, a hoax can often be perpetuated on holidays or other special days and could be carried out simply for fun. Regardless, hoaxes can use up valuable organization resources: email replies, Internet bandwidth used, time spent, and so on. An example of a "harmless" hoax was Google's supposed name change to "Topeka" on April Fools' Day 2010. An example of a financially harmful hoax was the supposed assassination of Bill Gates on April Fools' Day 2003. This hoax led to stock market fluctuations and loss of profit in Asia. Some companies place a time limit on jokes and hoaxes indicating that the affected person has become nonproductive; for example, 3 percent of the workday.

Pretexting, malicious insider attempts, diversion theft, phishing, and hoaxes are all known as *confidence tricks*, thus the term *con*, and are committed by "bunko" artists. However, there are even lower ways to get access to people's information; these often are used with the previous methods. They include

shoulder surfing, eavesdropping, dumpster diving, baiting, and piggybacking.

Impersonation or Pretexting

Pretexting is a type of social engineering attack in which a person invents a scenario, or pretext, in the hope of persuading a victim to divulge information. Preparation and some prior information are often needed before attempting a pretext; impersonation is often a key element. By impersonating the appropriate personnel or third-party entities, a person performing a pretext hopes to obtain records about an organization, its data, and its personnel. IT people and employees should always be on the lookout for impersonators and always ask for identification. If there is any doubt, the issue should be escalated to your supervisor and/or a call should be made to the authorities.

Eavesdropping

Eavesdropping is a strategy in which a person uses direct observation to "listen" in to a conversation. This could be a person hiding around the corner or a person tapping into a phone conversation. Soundproof rooms are often employed to stop eavesdropping, and encrypted phone sessions can also be implemented.

Baiting

Baiting is a strategy in which a malicious individual leaves malware-infected removable media such as a USB drive or optical disc lying around in plain view. It might have an interesting logo or distinctive look about it. When a person takes it and connects it to his or her computer, the malware infects the computer and attempts to take control of it and/or the network the computer is a member of.

Watering Hole Attack

The ***watering hole attack*** is a strategy that targets users based on the common websites that they frequent. The attacker loads malware beforehand on one or more websites in the hopes that the user will access those sites and activate the malware, ultimately infecting the user's system and possibly spreading through the network. To figure out the browsing habits of users, the attacker might guess or use direct observation. So, this attack may also build on other social engineering methods such as eavesdropping, pretexting, and phishing.

Popular websites such as Google, Microsoft, and so on will be difficult to infect with malware. It's the smaller websites that the attacker will go after. For example, let's consider a company that manufactures widgets. Chances are that the company will need to purchase plastic and other resources to build the widgets. It follows that users will connect to suppliers' websites often via the Internet or possibly an intranet. Typically, suppliers' websites are known for a lack of security and make excellent targets. If many users in the company go to these same websites, and often, it's just a matter of time before one clicks on the wrong website element or gets tricked in another manner. Then malware gets installed to the client computer and possibly spreads throughout the company. An attacker might also redirect users to other websites where more hardcore malware (such as ransomware) or other scams are located.

The problem is that you, as a security administrator, can't actively prevent the malware on the targeted websites. You can suggest prevention methods to those companies—such as software patches and secure coding—but can't force them into action. So, you should focus on localized prevention methods including training users, reducing web browser functionality, blocking known malicious websites based on threat intelligence, and monitoring in the form of antimalware software, IDS/IPS, and more—essentially.

Typo Squatting

Typo squatting (or *typosquatting*) is a technique used by adversaries that

leverages human error when typing a URL in their web browser. For instance, users often enter typos (or a wrong URL) for a given website. Let's suppose you want to go to google.com, but instead typed gogle.com. An attacker can register the "gogle.com" domain and impersonate the real website and host malware or perform any other malicious technique to compromise your system.

Influence Campaigns, Principles of Social Engineering, and Reasons for Effectiveness

The following are several motivation techniques used by social engineers:

- **Authority:** A social engineer shows confidence and perhaps authority—whether legal, organizational, or social authority.
- **Intimidation:** Attackers can use intimidation to manipulate their victims to perform some action or to reveal sensitive information.
- **Consensus:** Consensus, or "social proof," is a psychological phenomenon in which an individual is not able to determine the appropriate mode of behavior. For example, you might see others acting or doing something in a certain way and might assume that it is appropriate. Social engineers might use this tactic when an individual enters an unfamiliar situation that he or she doesn't know how to deal with. Social engineers might manipulate multiple people at once by using this technique.
- **Scarcity:** It is possible to use scarcity to create a feeling of urgency in a decision-making context. Specific language can be used to heighten urgency and manipulate victims. Salespeople often use scarcity to manipulate clients (for example, telling a customer that an offer is for today only or that there are limited supplies). Social engineers use similar techniques.
- **Familiarity:** Individuals can be influenced by things or people they like or are familiar with. Social engineers take advantage of these human

vulnerabilities to manipulate their victims.

- **Trust:** Attackers take advantage of the trust a person has in another person or organization in order to influence them to perform some action or reveal sensitive information.
- **Urgency:** It is possible to manipulate a person with a sense of immediate urgency to prompt him or her to act promptly. Using urgency, social engineers force their victims to act quickly to avoid or rectify a perceived dangerous or painful situation.

Tip

Nowadays, threat actors use automated bots in ***social media*** sites like Twitter and Facebook to influence the sentiment of a given user. These bots are used to try to manipulate public sentiment on contentious issues including political events, gun control, abortion, and many more. ***Hybrid warfare*** is a subject that was originally employed by the armed forces. However, attackers nowadays use hybrid warfare techniques in cyber and influential campaigns to manipulate people to believe something that may not be true by using different types of propaganda that is often shared in social media sites.

User Security Awareness Education

The user could be the single most exploitable resource of a company. I think that the previous section about social engineering provides a good argument to support that opinion. People aren't computers. Some might be compared to robots (perhaps unfairly), but no matter how logical and disciplined a person might be, there is always the potential for human error. So, until firewalls are developed for people's brains, education becomes the best method to prevent social engineering attacks and malware infection, not to mention user error.

As an IT manager, I have always made it a point to incorporate technology

and security training for as many employees as possible. This would include classroom/conference room training, written materials, digital courses, even funny posters in the cafeteria. I'd use whatever platform I could to get the point across. The key is to make it accessible to the user and to make it interesting and fun.

There are several roadblocks when it comes to user training. The first is organizational acceptance. Are the executives of a company on board with the idea? As time moves on, we see that more and more executives include user education as a matter of course. However, if you come across an individual who is against the idea because of a "lack" of budgeting or time, you can simply show that person a news article about one of the many successful attacks that have occurred recently. Then show a case study of the amount of time and money that the affected company lost due to the attack, and—in most cases—how easily it could have been prevented. Then, there are the employees to be trained themselves. Some will put up a fight when it comes to education. Again, the secret ingredient here is to pique the interest of the users. Get them involved, make it fun, create a reward system, and use your imagination. Some organizations employ IT trainers on a full-time basis or as consultants. Good IT trainers know how to get through to the typical employee. Other organizations will offer incentives for attending training, or penalties for not attending—though statistics show that incentives usually work better than penalties.

You can also attempt role-playing. I'm not talking about a role-playing game such as *Dungeons & Dragons*! Rather, having the employees act out different organizational roles, such as system administrators, privileged users, executive users, data owners, system owners, and of course, typical end users. Throw in the hacker and/or con artist as the "bad guy" roles and you can really teach in a fun way. Create scenarios where people can learn in a tangible way how attacks are carried out and how they can be prevented. Quiz the employees, but keep it light. The idea is for employees to expand their knowledge into other areas of the company—a sort of table-top job rotation, so to speak. Develop the situations and solutions properly, and the whole organization becomes a stronger and more secure unit, thanks to you. Come on, you know you always aspired to be a Dungeon Master!

Finally, there is the time factor. People have projects and tasks to complete and usually aren't even given enough time for that! Where does the time for

training come from? This is when the mindset of loyalty to the company comes in—and human resources can usually be helpful in cultivating that mindset. The whole outlook should be based on the idea of overall efficiency and benefits to the company and individual. By sitting in on security training, the user will save time over the long term and will ultimately become a more knowledgeable person.

Anyway, for the Security+ certification, *how* the training gets accomplished isn't as important as *what* is covered in training. The following is a basic list of rules you can convey when training employees:

- Never, under any circumstances, give out any authentication details such as passwords, PINs, company ID, tokens, smart cards, and so on.
- Always shield keypads and screens when entering authentication information.
- Adhere to the organization's *clean desk policy*, which states that all documents, electronics, personally owned devices, and other items be put away (or locked away) when you are not at your desk, or other work area.
- Always screen your email and phone calls carefully and keep a log of events. This is also known as communications *vetting*.
- Use encryption when possible to protect emails, phone calls, and data.
- If there is any doubt as to the legitimacy of a person, email, or phone call, document the situation and escalate it to your supervisor, security, or the authorities.
- Never pick up, and make use of, any unknown removable media.
- Always shred any sensitive information destined for the garbage or recycling.
- Always comply with company policy when it comes to data handling and disposal. For example, if a hard drive, USB flash drive, memory stick, or optical disc is no longer being used, make sure it is disposed of properly. If you are not sure, contact the IT department or facilities

department of the organization to find out if it should be recycled, or destroyed.

- Always track and expedite shipments.
- Be extremely careful when using a web browser. Double-check everything that is typed before pressing Enter or clicking Go. Don't click on anything unless you know exactly what it is. Web-based attacks account for a huge percentage of damage to organizations.

When training employees, try to keep them interested; infuse some fun and be silly if you want to. For instance, the first bullet said to *never* give out authentication details. Pundits tell us to never say never. Well, if it's okay for them, then it's okay for us. And in this case, it's vital to the health of your organization—not to mention you. You see what I mean? Or, if you don't want to be silly, then consider imparting some real examples. Use examples of social engineering so that your trainees can make the connection between actual social engineering methods and their defenses. Make them understand that social engineers don't care how powerful an organization's firewall is or how many armed guards the company has. They get past technology and other types of security by exploiting the weaknesses inherent in human nature.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 1-2 lists a reference of these key topics and the page number on which each is found.

Table 1-2 Key Topics for Chapter 1

Key Topic Element	Description	Page Number
Section	Defining and understanding what phishing and spear phishing are	
Paragraph	Defining and understanding what smishing is	
Paragraph	Defining and understanding what vishing is	
Paragraph	Defining and understanding what spam and spam over Internet messaging (SPIM) is	
Paragraph	Defining and understanding what dumpster diving is	
Paragraph	Defining and understanding what shoulder surfing is	
Figure 1-1	Defining and understanding what pharming is	
Paragraph	Defining and understanding what tailgating is	
Paragraph	Defining and understanding what eliciting information is	
Paragraph	Defining and understanding what whaling is	
Paragraph	Defining and understanding what	

	prepending is	
Paragraph	Defining and understanding what identity fraud is	
Paragraph	Defining and understanding what invoice scams are	
Paragraph	Defining and understanding what credential harvesting is	
Paragraph	Defining and understanding what reconnaissance is	
Paragraph	Defining and understanding what a hoax is	
Paragraph	Defining and understanding what impersonation is	
Paragraph	Defining and understanding what a watering hole attack is	
Paragraph	Defining and understanding what typo squatting is	
Paragraph	Surveying influence campaigns and understanding the principles of social engineering and reasons for effectiveness	
Paragraph	Understanding the importance of user security awareness education	
List	Rules you can convey when training employees	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

phishing

pretexting

spear phishing

whaling

smishing

vishing

spam

dumpster diving

shoulder surfing

pharming

piggybacking

tailgating

eliciting information

prepending

credential harvesting

hoax

watering hole attack

typo squatting

social media

hybrid warfare

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. You go out the back door of your building and notice someone looking through your company's trash. If this person were trying to acquire sensitive information, what would this attack be known as?
2. User education can help to defend against which type of attacks?
3. You can often configure your email servers or email cloud services to prepend a message in the email subject line to identify emails that are coming from outside of the organization. This technique is also known as _____.
4. What is the most common reason that social engineering succeeds?
5. In which two environments would social engineering attacks be most effective?
6. What is a social engineering technique used by adversaries that leverages user errors ("typos") when entering a given URL in their web browser for a given website. An attacker can register the "misspelled" domain to impersonate the real website, host malware, or perform any other malicious technique to compromise the end user's system.
7. A man pretending to be a data communications repair technician enters your building and states that there is networking trouble and he needs access to the server room. What is this an example of?
8. Turnstiles, double entry doors, and security guards are all preventive measures for what kind of social engineering?
9. What is the social engineering technique where the attacker redirects the victim from a valid website or resource to a malicious one that could be made to appear as the valid site to the user?
10. Why would you implement password masking and privacy screens/filters?

Chapter 2. Analyzing Potential Indicators to Determine the Type of Attack

This chapter covers the following topics related to Objective 1.2 (Given a scenario, analyze potential indicators to determine the type of attack.) of the CompTIA Security+ SY0-601 certification exam:

- Malicious Software (Malware)
    - Ransomware
    - Trojans
    - Worms
    - Potentially unwanted programs (PUPs)
    - Fileless virus
    - Command and control
    - Bots
    - Cryptomalware
    - Logic bombs
    - Spyware
    - Keyloggers
    - Remote access Trojan (RAT)
    - Rootkit
    - Backdoor
- Password attacks
    - Spraying

    - Dictionary
    - Brute force (Offline and Online)
    - Rainbow tables
    - Plaintext/unencrypted
- Physical attacks
    - Malicious universal serial bus (USB) cable
    - Malicious flash drive
    - Card cloning
    - Skimming
- Adversarial artificial intelligence (AI)
    - Tainted training data for machine learning (ML)
    - Security of machine learning algorithms
- Supply-chain attacks
- Cloud-based vs. on-premises attacks
- Cryptographic attacks
    - Birthday
    - Collision
    - Downgrade

To combat the various security threats that can occur on a computer system, you first need to classify them. Then you need to define how these threats can be delivered to the target computer. Afterward you can discuss how to prevent security threats from happening and troubleshoot them if they do occur. In this chapter, let's start with the most common computer threat and probably the deadliest—malicious software.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review

Activities ” section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 2-1 lists the major headings in this chapter and their corresponding “Do I Know This Already?” quiz questions. You can find the answers in Appendix A, “Answers to the ‘Do I Know This Already?’ Quizzes and Review Questions.”

Table 2-1 “Do I Know This Already?” Section-to-Question Mapping

Foundation Topics Section	Questions
Malicious Software (Malware)	1–5
Password Attacks	6–7
Physical Attacks	8
Adversarial Artificial Intelligence (AI)	9
Supply-Chain Attacks	10
Cloud-based vs. On-premises Attacks	11
Cryptographic Attacks	12

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. You logged in to your laptop and noticed a message saying that all your files have been encrypted and to recover them you need to pay $1,000 in

Bitcoins. What has your system been infected with?

a. Ransomware

b. Worm

c. Keylogger

d. None of these answers are correct.

2. What type of malware can look like legitimate software but then performs negative actions to manipulate your system?

a. Trojan

b. Ransomware

c. Worm

d. None of these answers are correct.

3. Which malware type can allow an attacker to gain administrator privileges?

a. Keylogger

b. Rootkit

c. Ransomware

d. All of these answers are correct.

4. What type of malware does not reside on the hard drive of a computer?

a. Ransomware

b. Botnets

c. Fileless malware

d. None of these answers are correct.

5. Bots in a botnet typically receive instructions from which of the following?

a. A command-and-control (C2) server

b. A zombie system

c. A malvertising site

d. All of these answers are correct.

6. An attacker using John the Ripper, which uses a wordlist, is an example of which of the following?

a. Social engineering attack

b. Dictionary password attack

c. Buffer overflow attack

d. Cross-site request forgery attack

7. An attacker using a large number of usernames with a few commonly used passwords is considered what kind of attack?

a. Password spraying

b. Credential harvesting

c. Password cracking

d. None of these answers are correct.

8. What type of attack occurs when an attacker captures credit card information or information from other similar cards (gift cards, loyalty cards, identification cards, and so on)?

a. Skimming

b. Shimming

c. SIM cloning

d. None of these answers are correct.

9. Which of the following techniques are used to attack machine learning (ML) implementations?

a. Tainting of data to cause errors in the outcome of the ML solution

b. Overfitting attacks

c. ML transfer attacks

d. All of these answers are correct.

10. You purchased a brand-new Internet of Things (IoT) device and noticed that it started collecting personal information (PI) and attempted to send

your data by communicating with random IP addresses. You noticed that an implant could have been installed during the manufacturing of the product. What type of attack might this be?

a. Supply-chain attack

b. Cross-site scripting

c. Return to libc attack

d. Masquerading attack

11. Which of the following attacks are made against cloud implementations?

a. API attacks

b. DNS attacks

c. VM escape attacks

d. All of these answers are correct.

12. An attacker attempts to force an application to roll back the version of TLS (from TLS version 1.3 to 1.0). What is the name of this type of attack?

a. Privilege escalation

b. Downgrade attack

c. Cracking

d. Fuzzing

Foundation Topics

Malicious Software (Malware)

Malicious software, or ***malware***, is software designed to infiltrate a computer system and possibly damage it without the user's knowledge or consent. *Malware* is a broad term used by computer professionals to include viruses,

worms, Trojan horses, spyware, rootkits, adware, and other types of undesirable software.

Of course, as a security professional, you don't want malware to infect your systems, but to defend against it, you first need to define it and categorize it. Then you can put preventive measures into place. It's also important to locate and remove or quarantine malware from a computer system in the case that it does manifest itself.

For the exam, you need to know about several types of malware. Over the past several years, an emphasis shift from viruses to other types of malware, such as spyware and ransomware, has occurred. Most people know about viruses and have some kind of antivirus software running. However, many people are still confused about the various other types of malware, how they occur, and how to protect against them. As a result, computer professionals spend a lot of time fixing malware issues (that are not virus related) and training users on how to protect against them in the future. However, viruses are still a valid foe, so let's start by discussing them.

A computer virus is code that runs on a computer without the user's knowledge; it infects the computer when the code is accessed and executed. For viruses to do their dirty work, they first need to be executed by the user in some way. A virus also has reproductive capability and can spread copies of itself throughout the computer—if it is first executed, by the user or otherwise. By infecting files accessed by other computers, the virus can spread to those other systems as well. The problem is that computers can't call in sick on Monday; they need to be up and running as much as possible, more than the average human.

Ransomware and Crypto Malware

Some less than reputable persons use a particularly devious malware known as ***ransomware***—a type of malware that restricts access to a computer system and demands that a ransom be paid. Ransomware informs the user that in order to decrypt the files or unlock the computer to regain access to the files, a payment would have to be made to one of several banking services

(typically crypto currencies like Bitcoin). It often propagates as a Trojan or worm, and usually makes use of encryption to cause the user's files to be inaccessible. This use of encryption is also known as cryptoviral extortion. Examples of ransomware include WannaCry, NotPetya, Nyetya, SamSam, BadRabbit, and others. One of the most common infection methods of ransomware is via phishing and spear phishing attacks. You learned about phishing and spear phishing in Chapter 1, "Comparing and Contrasting Different Types of Social Engineering Techniques."

Ransomware is also often referred to as ***cryptomalware***. Today's cryptomalware leverages advanced encryption techniques to prevent files from being decrypted without a unique key.

Figure 2-1 shows a message displayed to the victim of a ransomware attack. The ransomware illustrated in this figure is the well-known ransomware called WannaCry.

Figure 2-1 A Ransomware Attack

Note

Sometimes a user will inadvertently access a fraudulent website (or pop-up site) saying that all the user's files have been encrypted and payment is required to decrypt them; some

imposing government-like logo will accompany the statement. But many of these sites don't actually encrypt the user's files. In this case, we are talking about plain old extortion with no real damage done to the computer or files. These types of sites can be blocked by pop-up blockers, phishing filters, and the user's common sense when clicking searched-for links.

Trojans

Trojan horses, or simply ***Trojans***, appear to perform desirable functions but are actually performing malicious functions behind the scenes. They are not technically viruses and can easily be downloaded without being noticed. They can also be transferred to a computer by way of removable media, especially USB flash drives. One example of a Trojan is a file that is contained within a downloaded program such as a key generator—known as a keygen used with pirated software—or another executable. If users complain about slow system performance and numerous antivirus alerts, and they recently installed a questionable program from the Internet or from a USB flash drive, their computers could be infected by a Trojan.

Remote Access Trojans (RATs) and Rootkits

Remote access Trojans (RATs) are the most common type of Trojan. Examples include Back Orifice, NetBus, and SubSeven. Their capability to allow an attacker higher administration privileges than those of the owner of the system makes them quite dangerous. The software effectively acts as a remote administration tool, which happens to be another name for the *RAT* acronym. These programs have the capability to scan for unprotected hosts and make all kinds of changes to a host when connected. They are not *necessarily* designed to be used maliciously but are easy for an average person to download and manipulate computers with. Worse, when a target

computer is controlled by an attacker, it could easily become a robot, or simply a *bot*, carrying out the plans of the attacker on command. We discuss bots later in this chapter.

RATs can also be coded in many programming languages (such as PHP, Ruby, and Python) to allow remote access to websites. An example of this is the web shell, which has many permutations. It allows an attacker to remotely configure a web server without the user's consent. Quite often, the attacker will have cracked the FTP password in order to upload the RAT.

RATs are often used to persistently target a specific entity such as a government or a specific corporation. One example is the PlugX RAT. Malicious software such as this is known as an advanced persistent threat (APT). Groups that have vast resources at their disposal might make use of these APTs to carry out objectives against large-scale adversaries. APT groups could take the form of large hacker factions, and even some corporations and governments around the globe.

A ***rootkit*** is a type of software designed to gain administrator-level control over a computer system without being detected. The term is a combination of the words *root* (meaning the root user in a UNIX/Linux system or administrator in a Windows system) and *kit* (meaning software kit). Usually, the purpose of a rootkit is to perform malicious operations on a target computer at a later date without the knowledge of the administrators or users of that computer. A rootkit is a variation on the virus that attempts to dig in to the lower levels of the operating system—components of the OS that start up before any antimalware services come into play. Rootkits can target the UEFI/BIOS, boot loader, kernel, and more. An example of a boot loader rootkit is the Evil Maid Attack, which can extract the encryption keys of a full disk encryption system, as discussed later. Another (more current) example is the Alureon rootkit, which affects the master boot record (MBR) and low-level system drivers (such as atapi.sys). This particular rootkit was distributed by a botnet and affected over 200,000 (known) Microsoft operating systems.

Rootkits are difficult to detect because they are activated before the operating system has fully booted. A rootkit might install hidden files, hidden

processes, and hidden user accounts. Because rootkits can be installed in hardware or software, they can intercept data from network connections, keyboards, and so on.

A successfully installed rootkit enables unauthorized users to gain access to a system and act as the root or administrator user. Rootkits are copied to a computer as a binary file. This binary file can be detected by signature-based and heuristic-based antivirus programs; however, after the rootkit is executed, it can be difficult to detect. The reason is that most rootkits are collections of programs working together that can make many modifications to the system. When subversion of the operating system takes place, the OS can't be trusted, and it is difficult to tell if your antivirus programs run properly or if any of your other efforts have any effect. Although security software manufacturers are attempting to detect running rootkits, it is doubtful that they will be successful. The best way to identify a rootkit is to use removable media (a USB flash drive or a special rescue disc) to boot the computer. This way, the operating system is not running, and therefore, the rootkit is not running, making it much easier to detect by the external media.

Sometimes, rootkits will hide in the MBR. Often, operating system manufacturers recommend scrubbing the MBR (rewriting it, for example, within System Recovery Options or another recovery environment) and then scanning with antivirus software. This solution depends on the type of rootkit. The use of GPT in lieu of MBR helps to discourage rootkits. I suggest using GPT whenever possible.

Unfortunately, because of the difficulty involved in removing a rootkit, the best way to combat rootkits is to reinstall all software (or reimage the system). Generally, an IT technician, upon detecting a rootkit, will do just that because reinstalling usually takes less time than attempting to fix all the rootkit issues; plus, it can verify that the rootkit has been removed completely.

Worms

A ***worm*** is much like a virus except that it self-replicates, whereas a virus

does not. It does this in an attempt to spread to other computers. Worms take advantage of security holes in operating systems and applications, including backdoors, which we discuss later. They look for other systems on the network or through the Internet that are running the same applications and replicate to those other systems. With worms, the user doesn't need to access and execute the malware. A virus needs some sort of carrier to get it where it wants to go and needs explicit instructions to be executed, or it must be executed by the user. The worm does not need this carrier or explicit instructions to be executed.

Going back in history, a well-known example of a worm is Nimda (*admin* backward), which propagated automatically through the Internet in 22 minutes in 2001, causing widespread damage. It spread through network shares, mass emailing, and operating system vulnerabilities.

Sometimes, the worm does not carry a payload, meaning that in and of itself, it does not contain code that can harm a computer. It may or may not include other malware, but even if it doesn't, it can cause general disruption of network traffic and computer operations because of the very nature of its self-replicating abilities.

In the late 2010s, different types of ransomware (including WannaCry) propagated like a worm by scanning and infecting vulnerable systems leveraging a known Windows SMB vulnerability.

Fileless Virus

Malware doesn't have to reside on the hard drive of a computer. It can also reside within RAM (and possibly other locations). ***Fileless malware***—also known as non-malware—functions without putting malicious executables within the file system and instead works in a memory-based environment. Sound potent? It can be, but the attacker will usually need to have remote access to the system—via SSH, a RAT, or otherwise—in order to deploy the malware. The term *fileless* is somewhat of a misnomer because the attack may actually contain files. The systems affected, such as ATMs, often suffer from weak endpoint security, and the organization that owns those systems

might also have ineffective access control policies.

Command and Control, Bots, and Botnets

I know what you're thinking: the names of these attacks and delivery methods are getting a bit ridiculous. But bear with me; they make sense and are deadly serious. Allow me to explain—malware can be distributed throughout the Internet by a group of compromised computers (***bots***), known as a ***botnet***, and controlled by a master computer (***command-and-control*** [C2] server). The individual compromised computers in the botnet are called bots and also often called zombies because they are unaware of the malware that has been installed on them. These compromises can occur in several ways, including automated distribution of the malware from one zombie computer to another. Now imagine if all the zombie computers had a specific virus or other type of attack loaded, and a logic bomb (defined a bit later) was also installed, ready to set off the malware at a specific time. If this were done to hundreds or thousands of computers, a synchronized attack of great proportions could be enacted on just about any target. Often, this is known as a distributed denial-of-service (DDoS) attack and is usually perpetuated on a particularly popular server, one that serves many requests. If a computer on your network is continually scanning other systems on the network, is communicating with an unknown server, and/or has hundreds of outbound connections to various websites, chances are the computer is part of a botnet. But botnets can be used for more than just taking down a single target. They can also be used to fraudulently obtain wealth and even crypto mining.

Figure 2-2 shows a botnet controlled by a command-and-control server. The attacker (via the C2) sends instructions to the bots in the botnet. A botnet could be composed of any compromised system. These could be user endpoints, as well as Internet of Things (IoT) devices (such as cameras, sensors, and industrial control systems).

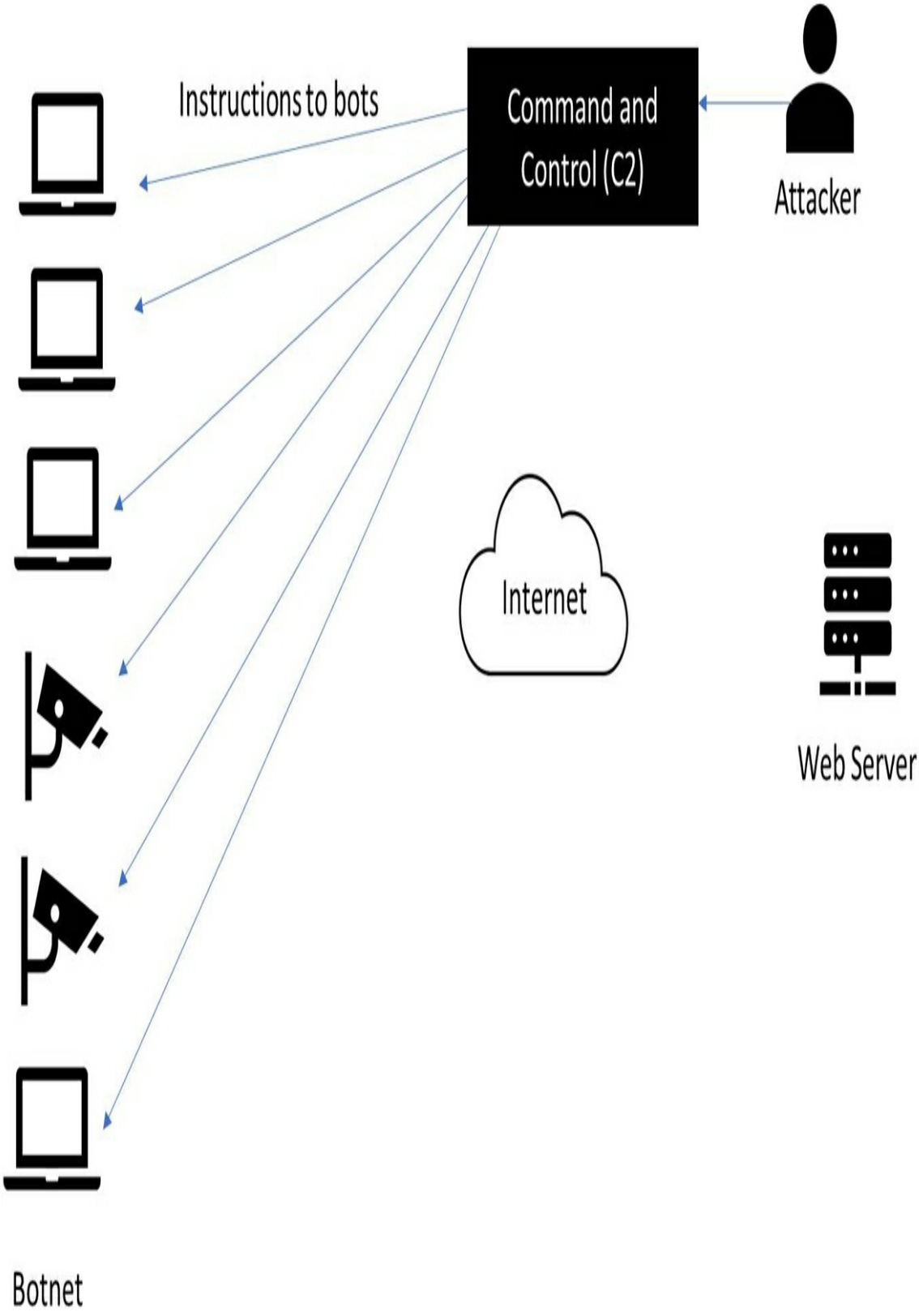

Instructions to bots
Command and Control (C2)
Attacker
Internet
Web Server
Botnet

Figure 2-2 A Botnet and C2 Example

After the bots receive the instructions from the C2, they perform a given action. In the example illustrated in Figure 2-3, the bots start sending numerous IP packets to a web server (the victim) to perform a DDoS attack. The purpose of this attack is to consume system resources and prevent authorized users from accessing the web server. You will learn additional details about DDoS attacks in Chapter 4, "Analyzing Potential Indicators Associated with Network Attacks."

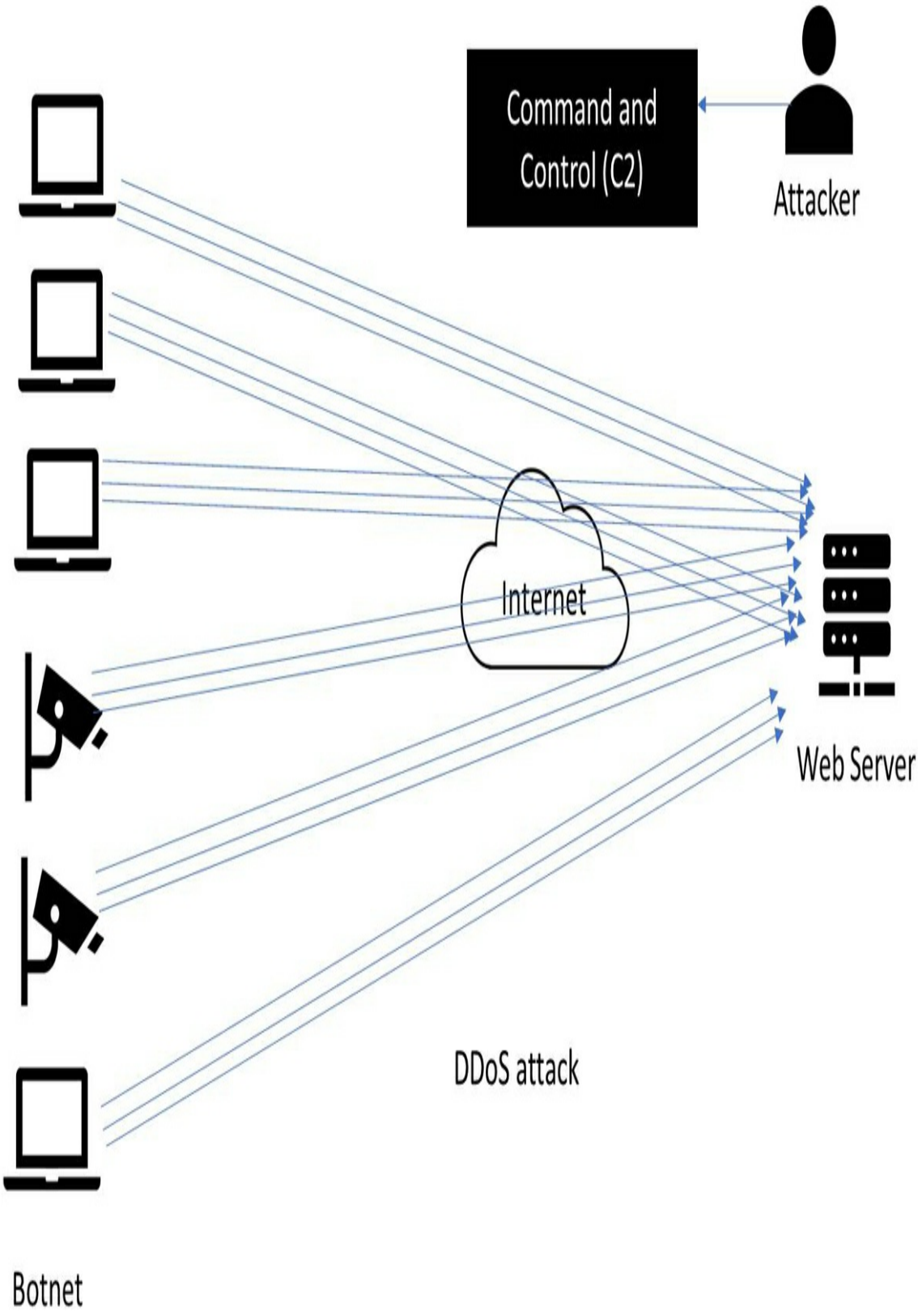

Command and Control (C2)
Attacker
Internet
Web Server
DDoS attack
Botnet

Figure 2-3 The Botnet Performing a DDoS Attack

Logic Bombs

A ***logic bomb*** is code that has, in some way, been inserted into software; it is meant to initiate one of many types of malicious functions when specific criteria are met. Logic bombs blur the line between malware and a malware delivery system. They are indeed unwanted software but are intended to activate viruses, worms, or Trojans at a specific time. Trojans set off on a certain date are also referred to as *time bombs*. The logic bomb ticks away until the correct time, date, and other parameters have been met. So, some of the worst bombs do not incorporate any explosion whatsoever. The logic bomb could be contained within a virus or loaded separately. Logic bombs are more common in the movies than they are in real life, but they do happen—and with grave consequences. But more often than not, they are detected before they are set off. If you, as a security administrator, suspect that you have found a logic bomb or a portion of the code of a logic bomb, you should notify your superior immediately and check your organization's policies to see if you should take any other actions. Action could include placing network disaster recovery processes on standby, notifying the software vendor, and closely managing usage of the software, including, perhaps, withdrawing it from service until the threat is mitigated. Logic bombs are the evil cousin of the Easter egg.

Easter eggs historically have been platonic extras added to an OS or application as a sort of joke; often, they were missed by quality control and subsequently released by the manufacturer of the software. Easter eggs are not normally documented (being tossed in last minute by humorous programmers) and are meant to be harmless, but nowadays they are not allowed by responsible software companies and are thoroughly scanned for. Because an Easter egg (and who knows what else) can possibly slip past quality control, and because of the growing concerns about malware in general, many companies have adopted the idea of *trustworthy computing*, which is a newer concept that sets standards for how software is designed, coded, and checked for quality control. Sadly, as far as software goes, the

Easter egg's day has passed.

Potentially Unwanted Programs (PUPs) and Spyware

Spyware is a type of malicious software either downloaded unwittingly from a website or installed along with some other third-party software. Usually, this malware collects information about the user without the user's consent. Spyware could be as simple as a piece of code that logs what websites you access, or it could go as far as a program that records your keystrokes (known as a keylogger). Spyware is also associated with advertising (those pop-ups that just won't go away!) and is sometimes related to malicious advertising, or *malvertising*—the use of Internet-based advertising (legitimate and illegitimate) to distribute malicious software.

Spyware can possibly change the computer configuration without any user interaction—for example, redirecting a browser to access websites other than those wanted. One example (of many) of spyware is the Internet Optimizer, which redirects Internet Explorer error pages out to other websites' advertising pages. Spyware can even be taken to the next level and be coded in such a way to hijack a user's computer. Going beyond this, spyware can be used for cyber espionage, as was the case with Red October, which was installed to users' computers when they unwittingly were redirected to websites with special PHP pages that exploited a Java flaw, causing the download of the malware.

Adware usually falls into the realm of spyware because it pops up advertisements based on what it has learned from spying on the user. *Grayware* is another general term that describes applications that are behaving improperly but without serious consequences. It is associated with spyware, adware, and joke programs. Very funny…not. Adware and grayware are types of ***potentially unwanted programs (PUPs)***.

Preventing spyware and PUPs works in much the same manner as preventing other types of malware when it comes to updating the operating system and using a firewall. Also, because spyware is as common as viruses, antivirus companies and OS manufacturers add antispyware components to their software. Here are a few more things you can do to protect your computer in the hopes of preventing spyware:

- Use (or download) and update built-in antispyware programs such as Windows Defender. Be sure to keep the antispyware software updated.
- Adjust web browser security settings. For example, disable (or limit) cookies, create and configure trusted zones, turn on phishing filters, restrict unwanted websites, turn on automatic website checking, disable scripting (such as JavaScript and ActiveX), and have the browser clear all cache on exit. All of these things can help filter out fraudulent online requests for usernames, passwords, and credit card information, which is also known as web-page spoofing. Higher security settings can also help fend off session hijacking, which is the act of taking control of a user session after obtaining or generating an authentication ID.
- Uninstall unnecessary applications and turn off superfluous services (for example, Remote Desktop services or FTP if they are not used).
- Educate users on how to surf the web safely. User education is actually the number one method of preventing malware! Access only sites believed to be safe, and download only programs from reputable websites. Don’t click OK or Agree to close a window; instead press Alt+F4 on the keyboard to close that window, or use the Task Manager to close out of applications. Be wary of file-sharing websites and the content stored on those sites. Be careful of emails with links to downloadable software that could be malicious.
- Consider technologies that discourage spyware. For example, use a browser that is less susceptible to spyware, and consider using virtual machines.
- Verify the security of sites you visit by checking the certificate or by simply looking for HTTPS in the URL.

Here are some common symptoms of spyware:

- The web browser’s default home page has been modified.

- A particular website comes up every time you perform a search.
- Excessive pop-up windows appear.
- The network adapter's activity LED blinks frequently when the computer shouldn't be transmitting data.
- The firewall and antivirus programs turn off automatically.
- New programs, icons, and favorites appear.
- Odd problems occur within windows (slow system, applications behaving strangely, and such).
- The Java console appears randomly.

To troubleshoot and repair systems infected with spyware, first disconnect the system from the Internet (or simply from the local-area network). Then try uninstalling the program from the Control Panel or Settings area of the operating system. Some of the less malicious spyware programs can be fully uninstalled without any residual damage. Be sure to reboot the computer afterward and verify that the spyware was actually uninstalled! Next, scan your system with the AV software to remove any viruses that might have infested the system, which might get in the way of a successful spyware removal. Again, in Windows, do this in the recovery environment (for example, Safe Mode) if the AV software offers that option.

Keyloggers

An attacker may use a ***keylogger*** to capture every keystroke of a user in a system and steal sensitive data (including credentials). There are two main types of keyloggers: keylogging hardware devices and keylogging software. A hardware (physical) keylogger is usually a small device that can be placed between a user's keyboard and the main system. Software keyloggers are dedicated programs designed to track and log user keystrokes.

Tip

Keyloggers are legal in some countries and designed to allow employers to oversee the use of their computers. However, recent regulations like the General Data Protection Regulation (GDPR) in the European Union (EU) have made keyloggers a very sensitive and controversial topic. Threat actors use keyloggers for the purpose of stealing passwords and other confidential information.

There are several categories of software-based keyloggers:

- **Kernel-based keylogger:** With this type of keylogger, a program on the machine obtains root access to hide itself in the operating system and intercepts keystrokes that pass through the kernel. This method is difficult both to write and to combat. Such keyloggers reside at the kernel level, which makes them difficult to detect, especially for user-mode applications that don't have root access. They are frequently implemented as rootkits that subvert the operating system kernel to gain unauthorized access to the hardware. This makes them very powerful. A keylogger using this method can act as a keyboard device driver, for example, and thus gain access to any information typed on the keyboard as it goes to the operating system.
- **API-based keylogger:** With this type of keylogger, compromising APIs reside inside a running application. Different types of malware have taken advantage of Windows APIs, such as GetAsyncKeyState() and GetForeground Window(), to perform keylogging activities.
- **Hypervisor-based keylogger:** This type of keylogger is effective in virtual environments, where the hypervisor could be compromised to capture sensitive information.
- **Web form–grabbing keylogger:** Keyloggers can steal data from web form submissions by recording the web browsing on submit events.
- **JavaScript-based keylogger:** Malicious JavaScript tags can be injected into a web application and then capture key events (for example, the onKeyUp() JavaScript function).
- **Memory-injection-based keylogger:** This type of keylogger tampers with the memory tables associated with the browser and other system

functions.

Backdoors

Backdoors are pieces of software, malware, or configuration changes that allow attackers to control a victim's system remotely. For example, a backdoor can open a network port on the affected system so that the attacker can connect and control the system.

A backdoor can be installed by an attacker to either allow future access or to collect information to use in further attacks. When the threat actor gains access to a system, the attacker usually wants future access so he or she can control the system and can carry out other attacks. The attacker wants that access to be really easy. Many backdoors are installed by users clicking something without realizing that the link that they clicked or the file that they opened is actually a threat. A backdoor can be implemented as a result of malware, a virus, or a worm.

Malware Delivery Mechanisms

Malware is not sentient (…not yet) and can't just appear out of thin air; it needs to be transported and delivered to a computer or installed on a computer system in some manner. This can be done in several ways. The simplest way would be for attackers to gain physical access to an unprotected computer and perform their malicious work locally. But because obtaining physical access can be difficult, this can be done in other ways, as shown in the upcoming sections. Attackers can use some of these methods to simply gain access to a computer, make modifications, and so on, in addition to delivering the malware.

The method that a threat uses to access a target is known as a threat vector. Collectively, the means by which an attacker gains access to a computer in order to deliver malicious software is known as an attack vector. Probably the most common attack vector is via software.

Malware can be delivered via software in many ways. A person who emails a

zipped file might not even know that malware also exists in that file. The recipients of the email will have no idea that the extra malware exists unless they have software to scan their email attachments for it. Malware could also be delivered via FTP. Because FTP servers are inherently insecure, it's easier than you might think to upload insidious files and other software. Malware is often found among peer-to-peer (P2P) networks and bit torrents. Great care should be taken by users who use these technologies. Malware can also be embedded within, and distributed by, websites through the use of corrupting code or bad downloads. Malware can even be distributed by advertisements. And of course, removable media can victimize a computer as well. Optical discs, USB flash drives, memory cards, and connected devices such as smartphones can easily be manipulated to automatically run malware when they are inserted into the computer. (This is when AutoPlay/AutoRun is not your friend!) The removable media could also have hidden viruses or worms and possibly logic bombs (discussed earlier) configured to set that malware off at specific times.

Potential attackers also rely on user error. For example, if a user is attempting to access a website but types the incorrect domain name by mistake, that user could be redirected to an altogether unwanted website, possibly malicious in nature. This type of attack is known as typo squatting or URL hijacking.

Note

You learned details about typo squatting in Chapter 1.

URL stands for Uniform Resource Locator, which is the web address that begins with http or https. The potential attacker counts on the fact that millions of typos are performed in web browsers every day. These attackers "squat" on similar (but not exact) domain names. Once the user is at the new and incorrect site, the system becomes an easy target for spyware and other forms of malware. Some browsers come with built-in security such as antiphishing tools and the capability to autocheck websites that are entered, but the best way to protect against this issue is to train users to be careful when typing domain names.

Another way to propagate malware is to employ automation and web attack-

based software “kits,” also known as exploit kits. Exploit kits are designed to run on web servers and are usually found in the form of PHP scripts. They target client computers’ software vulnerabilities that often exist within web browsers. One example of an exploit kit is the Blackhole exploit kit. It is used (and purchased) by potential attackers to distribute malware to computers that meet particular criteria, while the entire process is logged and documented.

Note

The automating of cybercrime, and the software used to do so, is collectively referred to as *crimeware*.

Active interception normally includes a computer placed between the sender and the receiver to capture and possibly modify information. If a person can eavesdrop on your computer’s data session, then that data can be stolen, modified, or exploited in other ways. Examples include session theft and man-in-the-middle (MITM) attacks. For more information on these attacks, see Chapter 4.

You Can’t Save Every Computer from Malware!

On a sad note, sometimes computers become so infected with malware that they cannot be saved. In this case, the data should be backed up (if necessary by removing the hard drive and slaving it to another system), and the operating system and applications reinstalled. The UEFI/BIOS of the computer should also be flashed. After the reinstall, the system should be thoroughly checked to make sure that there are no residual effects and that the system’s hard drive performs properly.

Password Attacks

There are many different ways that attackers can steal passwords, crack passwords, and perform other credential-based (password-based) attacks. The following sections describe the most common password-based attacks.

Dictionary-based and Brute-force Attacks

A ***dictionary-based attack*** pulls words from the dictionary or word lists to attempt to discover a user's password. A dictionary attack uses a predefined dictionary to look for a match between the encrypted password and the encrypted dictionary word. Many times, a dictionary attack will recover a user's password in a short period of time if simple dictionary words are used.

A hybrid attack uses a dictionary or a word list and then prepends and appends characters and numbers to dictionary words in an attempt to crack the user's password. These programs are comparatively smart because they can manipulate a word and use its variations. For example, take the word *password*. A hybrid password audit would attempt variations such as *1password*, *password1*, *p@ssword*, *pa44w0rd*, and so on. Hybrid attacks might add some time to the password-cracking process, but they increase the odds of successfully cracking an ordinary word that has had some variation added to it.

A ***brute-force attack*** uses random numbers and characters to crack a user's password. A brute-force attack on an encrypted password can take hours, days, months, or years, depending on the complexity and length of the password. The speed of success depends on the speed of the CPU's power. Brute-force audits attempt every combination of letters, numbers, and characters.

Tip

Tools such as 0phtCrack, LCP, Cain and Abel, and John the Ripper can all perform dictionary, hybrid, and brute-force password cracking.

Password Spraying

Password spraying is a type of attack in which an attacker attempts to compromise a system using a large number of usernames with a few commonly used passwords. Traditional brute-force attacks attempt to gain unauthorized access to a single account by guessing the password. This can quickly result in the targeted account getting locked out because commonly used account-lockout policies allow for a limited number of failed attempts (typically three to five) during a set period of time. During a password-spraying attack (also known as the "low-and-slow" method), the malicious actor attempts a single commonly used password (such as *Password1*, *COVID19*, or *asdasd*) against many accounts before moving on to attempt a second password, and so on. This technique allows the actor to remain undetected by avoiding rapid or frequent account lockouts.

Offline and Online Password Cracking

Weak passwords are the bane of today's operating systems and networks. This could be because no policy for passwords was defined, and people naturally gravitate toward weaker, easier-to-remember passwords. Or it could be that a policy was defined but is not complex enough or is out of date. Whatever the reason, it would be wise to scan computers and other devices for weak passwords with an ***online*** or ***offline password cracker***, which uses comparative analysis to break passwords and systematically guesses until it cracks the password. And, of course, a variety of password-cracking programs can help with this. Examples are John the Ripper and hashcat. These programs have a bit of a learning curve but are quite powerful. They can be used to crack all kinds of different passwords on a local system or on remote devices and computers. They sniff out other hosts on the network the way a protocol analyzer would. They are excellent tools to find out whether weak passwords are on the network or to help if users forget their passwords (when password resets are not possible). Attackers can discover hashed passwords on a compromised Windows, Linux, or macOS system and then

crack the password on the same system (online) or on a separate dedicated system (offline). The goal is to obtain the original plaintext version of the password.

Figure 2-4 shows how an attacker steals the password hash of a compromised system and then uses that hash to crack the password "offline" in another system with superior compute resources.

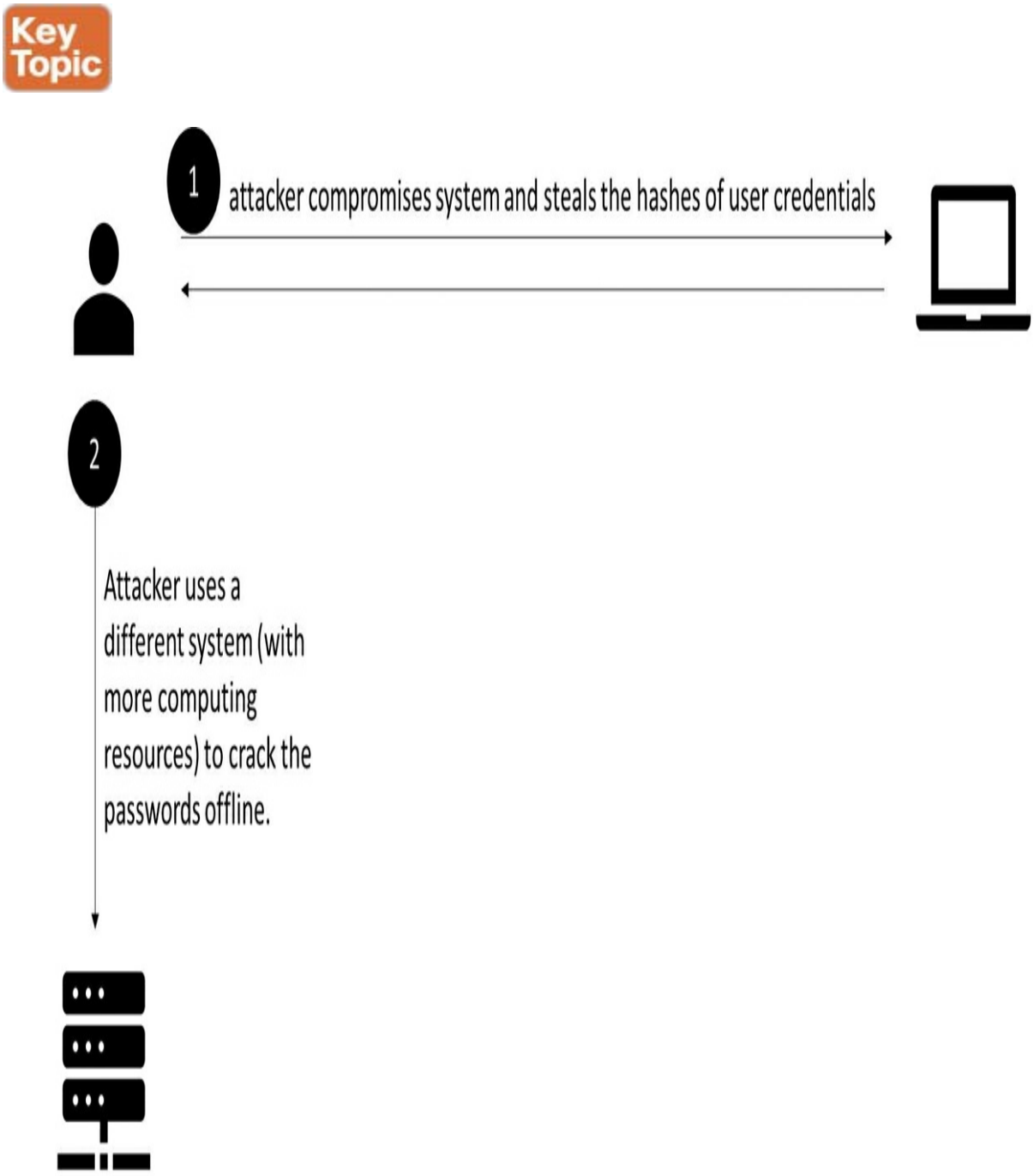


Figure 2-4 Offline Password Cracking

Rainbow Tables

A ***rainbow table*** is a precomputed table used to reverse engineer a cryptographic hash function and crack passwords. Attackers can perform cryptanalysis attacks a lot easier using these rainbow tables (as a form of a lookup table). Rainbow tables can be used with the L0phtCrack (https://sectools.org/tool/l0phtcrack/) and RainbowCrack applications.

> **Note**
>
> This attack can be defeated by implementing salting, which is the randomization of the hashing process.

Plaintext/Unencrypted

If your attempts to guess passwords have not been successful, sniffing or keystroke loggers might offer hope. Think about how much traffic passes over a typical network every day. Most networks handle a ton of traffic, and a large portion of it might not even be encrypted. Password sniffing requires that you have physical or logical access to the device. If that can be achieved, you can sniff the credentials right off the wire as users log in.

One such technique is to pass the hash. Passing the hash enables the hacker to authenticate to a remote server by using the underlying NT LAN Manager (NTLM) and/or LAN Manager (LM) hash of a user's password, instead of using the associated plaintext password. ***Plaintext*** is the term used to describe unencrypted data (in this case an unencrypted password). Mimikatz is a pass the hash application that enables an attacker to authenticate to a remote server using the LM/NTLM hash of a user's password, eliminating the need to

crack/brute-force the hashes to obtain the clear-text password. Because Windows does not salt passwords, they remain static in the Local Security Authority Subsystem Service (LSASS) from session to session until the password is changed. If the password is stored in LSASS and the attacker can obtain a password hash, it can be functionally equivalent to obtaining the clear-text password. Rather than attempting to crack the hash, attackers can simply replay them to gain unauthorized access. You can download this pass the hash toolkit at https://github.com/gentilkiwi/mimikatz.

Physical Attacks

An organization's building is one of its greatest assets and as such it should be properly protected. The following sections detail door access, biometric readers, access logs, and video surveillance to teach you some of the ways to protect the building, its contents, and its inhabitants and to ensure proper authentication when a person enters a building.

Malicious Flash Drives

Malware can be transferred to a computer by way of removable media, especially USB ***malicious flash drives***. For example, an attacker could install a Trojan or ransomware by using a malicious flash drive, if he or she has physical access to the targeted system. Alternatively, the attacker could place the USB flash drive somewhere and use some aspects of social engineering to fool the user into inserting it in his or her system and getting infected. For example, the attacker may attach a key chain or pictures of a dog or a cat. This approach makes the attack a little more personal. The victim may try to find out who those keys and USB flash drive belong to. When he or she inserts the USB flash into a machine, the system is compromised.

Malicious Universal Serial Bus (USB) Cables

In a similar way to malicious USB flash drives, attackers can use ***malicious USB cables*** to compromise systems. Different USB cables are designed to infect connected devices with malware. These malicious USB cables work by injecting keystrokes onto the victim's system when plugged into a USB-capable device.

Card Cloning Attacks

Attackers can perform different ***card cloning attacks***. For example, an attacker can clone a credit card, a smartphone SIM card, or even the badges/cards used to access a building. Specialized software and hardware can be used to perform these cloning attacks.

For instance, a traditional attack on smartphones is SIM cloning (also known as phone cloning), which allows two phones to utilize the same service and allows an attacker to gain access to all phone data. V1 SIM cards had a weak algorithm that made SIM cloning possible (with some expertise). However, V2 cards and higher are much more difficult (if not impossible) to clone due to a stronger algorithm on the chip. Users and administrators should be aware of the version of SIM card being used and update it (or the entire smartphone) if necessary.

There are techniques available to unlock a smartphone from its carrier. Users should be advised against this, and as a security administrator, you should create and implement policies that make unlocking the SIM card difficult if not impossible. Unlocking the phone—making it SIM-free—effectively takes it off the grid and makes it difficult to track and manage. When the SIM is wiped, the international mobile subscriber identity (IMSI) is lost, and afterward the user cannot be recognized. However, you can attempt to blacklist the smartphone through its provider by using the International Mobile Equipment Identity (IMEI), electronic serial number (ESN), or mobile equipment identifier (MEID). The ID used will vary depending on the type and age of smartphone. Regardless, as a security administrator, you

should avoid that tactic altogether because the damage has already been done, so protection of the SIM becomes vital.

Skimming

Skimming is a type of attack in which an attacker captures credit card information or information from other similar cards (gift cards, loyalty cards, identification cards, and so on) from a cardholder surreptitiously. Attackers use a device called a *skimmer* that can be installed at strategic locations such as ATMs and gas pumps to collect card data.

Attackers can also perform radio frequency identification (RFID) attacks to steal information. RFID has many uses, but it all boils down to identifying and tracking tags that are attached to objects. In the security world, that generally means authentication. As with any wireless technology, RFID is susceptible to attack. For example, some RFID tags can be affected by skimming, MITM attacks, sniffing, eavesdropping/replaying, spoofing, and jamming (DoS).

From an authentication standpoint, the attacker is using these attacks to try to find out the passcode. An RFID tag can also be reverse-engineered if the attacker gets possession of it. Finally, power levels can be analyzed to find out passwords. On some RFID tags, correct passcodes emit a different level of power than incorrect passcodes. To prevent these attacks, you, as the security administrator, or your team should consider newer-generation RFID devices, encryption, chip coatings, filtering of data, and multifactor authentication methods. Encryption is one of the best methods. Included in this prevention method are rolling codes, which are generated with a pseudorandom number generator (PRNG) and challenge-response authentication (CRA), where the user (or user's device) must present the valid response to the challenge.

RFID ranges vary depending on the EM band used—from 10 cm up to 200 meters. Many authentication readers can be as much as 1 meter, which is enough to facilitate skimming of information by attackers. One way to avoid this is to use newer RFID proximity readers and keys—ones that use lower

frequencies—from respectable companies.

Another way is to utilize the set of protocols called near-field communication (NFC). NFC generally requires that communicating devices be within 4 cm of each other, which makes skimming of information difficult. If an employee uses a smartphone to enter a building instead of an RFID device, then NFC should be implemented. NFC has obvious benefits for contactless payment systems or any other non-contact-oriented communications between devices. However, for optimal security, you should use contact-oriented readers and cards.

Adversarial Artificial Intelligence

Attackers have developed new ways to attack artificial intelligence (AI) and machine learning (ML) implementations. Examples of these adversarial techniques are tainting training data for ML and leveraging insecure implementations of machine learning algorithms.

Tainted Training Data for Machine Learning

Machine learning is being implemented in many solutions in use today (even in security products and solutions). Attackers can manipulate (taint) training data used for machine learning implementations to cause errors in the outcome of the ML solution. They can either change the integrity and modify the training data, or they can inject incorrect data in the training set.

Tip

If you are not familiar with the concepts of artificial intelligence and machine learning, the following article provides a good overview: https://vas3k.com/blog/machine_learning.

Security of Machine Learning Algorithms

Securing machine learning algorithms involves protecting the basis of security: confidentiality, integrity, and availability. The CIA concepts that apply to non-ML implementations also apply here with a few specific techniques. To protect ML implementations, you must classify malicious events in order to prevent them from interfering with algorithms and system operations. You need to prevent attackers from reaching ML system assets and from interfering with normal operations. You also need to prevent the attacker from reading critical data, as well as injecting any malicious or erroneous data in the training set.

Data poisoning attacks can be carried out by attackers to fool the machine learning system to produce incorrect errors. Another attack against machine learning implementations is the overfitting attack. When a machine learning system starts the learning process and "memorizes" its training data set, it will not generalize to new data. This process is referred to as overfitting. Overfitting and overfit models are very easy to attack because adversarial examples need only be a short distance away from the input to space in the ML system.

Many ML systems are implemented by tuning a base model that is already trained to enhance the learning process. ML transfer attacks can be launched by an attacker in this scenario. When the pretrained model is widely available, an attacker may be able to formulate attacks using this pretrained model that will be robust enough to succeed against your tuned task-specific model (which is typically unavailable to the attacker). In addition, the ML system you are fine-tuning could possibly be a Trojan injected by the attacker that includes devious ML behavior that is unanticipated.

Supply-Chain Attacks

A ***supply-chain attack*** is a type of attack in which attackers target security weaknesses in the supply network. Attackers have successfully launched supply-chain attacks against many different types of industries (including manufacturing plants, financial services companies, energy and oil companies, technology companies, and more). Attackers can modify products

or software during or right after the manufacturing process of a product by installing a rootkit or hardware-based spying components.

Cybersecurity experts recommend strict control of your supply network to prevent potential damage from today's attackers. The supply chain is a complex network of interconnected stakeholders governed by supply and demand.

Tip

An example of a supply-chain attack is the manipulation of system maintenance software called CCleaner. This software was manipulated in the supply chain including implants designed to spy and steal sensitive information from many large technology companies. You can obtain more information about this attack at https://blog.talosintelligence.com/2017/09/ccleaner-c2-concern.html.

Cloud-based vs. On-premises Attacks

Many organizations are moving to the cloud or deploying hybrid solutions to host their applications. Organizations moving to the cloud are almost always looking to transition from capital expenditure (CapEx) to operational expenditure (OpEx). Most Fortune 500 companies operate in a multicloud environment. It is obvious that cloud computing security is more important than ever. Cloud computing security includes many of the same functionalities as traditional IT security, which includes protecting critical information from theft, data exfiltration, and deletion, as well as privacy.

Note

You will learn details about cloud computing architectures in Chapter 10, "Summarizing Virtualization and Cloud Computing Concepts."

Cloud Security Threats

There are many potential threats when organizations move to a cloud model. For example, although your data is in the cloud, it must reside in a physical location somewhere. Your cloud provider should agree in writing to provide the level of security required for your customers. The following are questions to ask a cloud provider before signing a contract for its services:

- **Who has access?** Access control is a key concern because insider attacks are a huge risk. Anyone who has been approved to access the cloud is a potential hacker, so you want to know who has access and how they were screened. Even if it was not done with malice, an employee can leave, and then you find out that you don't have the password, or the cloud service gets canceled because maybe the bill didn't get paid.
- **What are your regulatory requirements?** Organizations operating in the United States, Canada, and the European Union must abide by many regulatory requirements (for example, ISO/IEC 27002, EU-U.S. Privacy Shield Framework, ITIL, and COBIT). You must ensure that your cloud provider can meet these requirements and is willing to undergo certification, accreditation, and review.
- **Do you have the right to audit?** This particular item is no small matter in that the cloud provider should agree in writing to the terms of the audit. With cloud computing, maintaining compliance could become more difficult to achieve and even harder to demonstrate to auditors and assessors. Of the many regulations touching on information technology, few were written with cloud computing in mind. Auditors and assessors might not be familiar with cloud computing generally or with a given cloud service in particular.

  Division of compliance responsibilities between cloud provider and cloud customer must be determined before any contracts are signed or service is started.
- **What type of training does the provider offer its employees?** This is a rather important item to consider because people will always be the

weakest link in security. Knowing how your provider trains its employees is an important item to review.

- **What type of data classification system does the provider use?** Questions you should be concerned with here include what data classification standard is being used and whether the provider even uses data classification.
- **How is your data separated from other users' data?** Is the data on a shared server or a dedicated system? A dedicated server means that your information is the only thing on the server. With a shared server, the amount of disk space, processing power, bandwidth, and so on is limited because others are sharing this device. If it is shared, the data could potentially become comingled in some way.
- **Is encryption being used?** Encryption should be discussed. Is it being used while the data is at rest and in transit? You will also want to know what type of encryption is being used. For example, there are big technical differences between DES and AES. For both of these algorithms, however, the basic questions are the same: Who maintains control of the encryption keys? Is the data encrypted at rest in the cloud? Is the data encrypted in transit, or is it encrypted at rest and in transit?
- **What are the service-level agreement (SLA) terms?** The SLA serves as a contracted level of guaranteed service between the cloud provider and the customer that specifies what level of services will be provided.
- **What is the long-term viability of the provider?** How long has the cloud provider been in business, and what is its track record? If it goes out of business, what happens to your data? Will your data be returned and, if so, in what format?
- **Will the provider assume liability in the case of a breach?** If a security incident occurs, what support will you receive from the cloud provider? While many providers promote their services as being "unhackable," cloud-based services are an attractive target to hackers.
- **What is the disaster recovery/business continuity plan (DR/BCP)?** Although you might not know the physical location of your services, it is physically located somewhere. All physical locations face threats such as fire, storms, natural disasters, and loss of power. In case of any of

these events, how will the cloud provider respond, and what guarantee of continued services is it promising?

Even when you end a contract, you must ask what happens to the information after your contract with the cloud service provider ends.

Insufficient due diligence is one of the biggest issues when moving to the cloud. Security professionals must verify that issues such as encryption, compliance, and incident response are worked out before a contract is signed.

Cloud Computing Attacks

Because cloud-based services are accessible via the Internet, they are open to any number of attacks. As more companies move to cloud computing, look for hackers to follow. Some of the potential attack vectors that criminals might attempt include the following:

- **Session hijacking:** This attack occurs when the attacker can sniff traffic and intercept traffic to take over a legitimate connection to a cloud service.
- **DNS attack:** This form of attack tricks users into visiting a phishing site and giving up valid credentials.
- **Cross-site scripting (XSS):** This type of attack is used to steal cookies that can be exploited to gain access as an authenticated user to a cloud-based service.
- **SQL injection:** This attack exploits vulnerable cloud-based applications that allow attackers to pass SQL commands to a database for execution.
- **Session riding:** This term is often used to describe a cross-site request forgery (CSRF) attack. Attackers use this technique to transmit unauthorized commands by riding an active session using an email or malicious link to trick users while they are currently logged in to a cloud service.
- **Distributed denial-of-service (DDoS) attack:** Some security professionals have argued that the cloud is more vulnerable to DDoS

attacks because it is shared by many users and organizations, which also makes any DDoS attack much more damaging.

- **Man-in-the-middle cryptographic attack:** This attack is carried out when the attacker places himself or herself in the communication path between two users. Anytime the attacker can do this, there is the possibility that he or she can intercept and modify communications.
- **Side-channel attack:** An attacker could attempt to compromise the cloud by placing a malicious virtual machine in close proximity to a target cloud server and then launching a side-channel attack.
- **Authentication attack:** Authentication is a weak point in hosted and virtual services and is frequently targeted. There are many ways to authenticate users, such as based on what a person knows, has, or is. The mechanisms used to secure the authentication process and the method of authentication used are frequent targets of attackers.
- **API attacks:** Often APIs are configured insecurely. An attacker can take advantage of API misconfigurations to modify, delete, or append data in applications or systems in cloud environments.

Regardless of the model used, cloud security is the responsibility of both the client and the cloud provider. These details will need to be worked out before a cloud computing contract is signed. The contracts will vary depending on the given security requirements of the client. Considerations include disaster recovery, SLAs, data integrity, and encryption. For example, is encryption provided end to end or just at the cloud provider? Also, who manages the encryption keys: the cloud provider or the client? Overall, you need to ensure that the cloud provider has the same layers of security (logical, physical, and administrative) in place that you would have for services you control.

Cryptographic Attacks

Attackers can launch different ***cryptographic attacks*** against weak cryptography (crypto) implementations. The following sections describe three common crypto attacks: collision, birthday, and downgrade attacks.

Collision

A ***collision*** occurs when two different files end up using the same hash. Message Digest Algorithm 5 (MD5) is a legacy hashing algorithm that is used to attempt to provide data integrity. By checking the hash produced by the downloaded file against the original hash, you can verify the file's integrity with a level of certainty. However, MD5 hashes are susceptible to collisions. Due to this low collision resistance, MD5 is considered to be harmful today. MD5 is also vulnerable to threats such as rainbow tables and preimage attacks. The best solution to protect against these attacks is to use a stronger type of hashing function such as SHA-2 or higher. The Secure Hash Algorithm (SHA) is one of a number of hash functions designed by the U.S. National Security Agency (NSA) and published by NIST. These functions are used widely in the U.S. government. Because MD5 and SHA-1 have vulnerabilities, government agencies and the private sector started using SHA-2 and newer implementations.

Note

You will learn more about hashing algorithms in Chapter 16, "Summarizing the Basics of Cryptographic Concepts."

It is important that a hashing algorithm be collision-resistant. If it has the capability to avoid the same output from two guessed inputs (by an attacker attempting a collision attack), it is collision-resistant. When it comes to cryptography, "perfect hashing" is not possible because usually unknowns are involved, such as the data to be used to create the hash and what hash values have been created in the past. Though perfect is not possible, it is possible to increase collision resistance by using a more powerful hashing algorithm.

Birthday

A ***birthday attack*** is an attack on a hashing system that attempts to send two different messages with the same hash function, causing a collision (similarly to the concept explained in the preceding section). It is based on the birthday problem in probability theory (also known as the birthday paradox). It can be summed up simply as the following: a randomly chosen group of people will have a pair of persons with the same calendar date birthday. Given a standard calendar year of 365 days, the probability of this occurring with 366 people is 100 percent (367 people on a leap year). So far, this makes sense and sounds logical.

The paradox (thoughtfully and mathematically) comes into play when fewer people are involved. With only 57 people, there is a 99 percent probability of a match (a much higher percentage than one would think), and with only 23 people, there is a 50 percent probability. Imagine that and blow out your candles! And by this, I mean use hashing functions with strong collision resistance. Because if attackers can find any two messages that digest the same way (use the same hash value), they can deceive a user into receiving the wrong message. To protect against a birthday attack, you should use a secure transmission medium, such as SSH, or encrypt the entire message that has been hashed.

Downgrade

A ***downgrade attack*** is a type of cryptographic attack that forces the rollback of a strong algorithm in favor of an older, lower-quality algorithm or mode of operation. Attackers leverage systems that have legacy crypto algorithms typically enabled for backward compatibility with older systems. Downgrade attacks can be performed by attackers in combination with an MITM attack.

Downgrade attacks can take many forms. However, they can target the crypto algorithm itself (such as downgrading from AES to DES or RC4) or the version of an algorithm (such as downgrading from TLS 1.3 to 1.0).

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 2-2 lists a reference of these key topics and the page number on which each is found.

Table 2-2 Key Topics for Chapter 2

Key Topic Element	Description	Page Number
Paragraph	Description of malware	
Paragraph	Description of ransomware	
Paragraph	Description of cryptomalware	
Paragraph	Description of Trojan horses	
Figure 2-1	A ransomware attack	
Paragraph	Description of remote access Trojans	
Paragraph	Description of rootkits	
Paragraph	Description of worms	
Paragraph	Description of fileless malware	
Paragraph	Description of Command and Control, bots, and botnets	
Figure 2-2	A botnet and C2 example	
Figure 2-3	The botnet performing a DDoS attack	
Paragraph	Description of logic bombs	
Paragraph	Description of spyware	
Paragraph	Description of potentially unwanted programs (PUPs)	
Paragraph	Description of keyloggers	
Paragraph	Description of backdoors	
Paragraph	Description of dictionary-based and brute-force attacks	
Paragraph	Description of password spraying	

Paragraph	Description of password cracking	
Figure 2-4	Offline password cracking	
Paragraph	Description of rainbow tables	
Paragraph	Description of password hacking via pass the hash technique	
Paragraph	Description of malicious flash drives	
Paragraph	Description of malicious USB drives	
Paragraph	Description of card cloning attacks	
Paragraph	Description of skimming	
Paragraph	Description of tainted training data used for machine learning	
Paragraph	Description of securing machine learning algorithms	
List	Questions to ask a cloud provider before signing a contract for its services	
List	Attack vectors against cloud-based services	
Paragraph	Description of collisions and hashing algorithm security vulnerabilities	
Paragraph	Description of birthday attacks	
Paragraph	Description of downgrade attacks	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

malware

ransomware

cryptomalware

Trojans

remote access Trojans (RATs)

rootkit

worm

fileless malware

bots

botnet

command and control

logic bomb

spyware

potentially unwanted programs (PUPs)

keylogger

backdoors

dictionary-based attack

brute-force attack

password spraying

online password cracker

offline password cracker

rainbow table

plaintext

malicious flash drives

malicious USB cables

card cloning attacks

skimming

supply-chain attack

cryptographic attacks

collision

birthday attack

downgrade attack

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What is a group of compromised computers that have software installed by a worm or Trojan?
2. What term is often used to describe a compromised system that can be updated automatically and remotely?
3. What is a common symptom of spyware?
4. You noticed that your DHCP server is flooded with information. After analyzing this condition, you found that the information is coming from more than 100 computers on the network. What is most likely the reason?

5. Which type of malicious software encrypts sensitive files and asks the user to pay in order to obtain a key recover those files?

6. What is a malicious attack that executes at the same time every week?

7. What is still one of the most common ways that attackers spread ransomware?

8. What is a type of malware that appears to a user as legitimate but actually enables unauthorized access to the user’s computer?

Chapter 3. Analyzing Potential Indicators Associated with Application Attacks

This chapter covers the following topics related to Objective 1.3 (Given a scenario, analyze potential indicators associated with application attacks) of the CompTIA Security+ SY0-601 certification exam:

- Privilege escalation
- Cross-site scripting
- Injections
    - Structured query language (SQL)
    - Dynamic link library (DLL)
    - Lightweight directory access protocol (LDAP)
    - Extensible markup language (XML)
- Pointer/object dereference
- Directory traversal
- Buffer overflows
- Race conditions (Time of check/time of use)
- Error handling
- Improper input handling
- Replay attack (session replays)
- Integer overflow
- Request forgeries
    - Server-side

    - Cross-site
- Application programming interface (API) attacks
- Resource exhaustion
- Memory leak
- Secure Socket Layer (SSL) stripping
- Driver manipulation
    - Shimming
    - Refactoring
- Pass the hash

Thanks to the advancements in modern web applications and related frameworks, the ways to create, deploy, and maintain web applications have changed such that the environment is now very complex and diverse. These advancements in web applications have also attracted threat actors.

Web-based applications are everywhere. You can find them for online retail, banking, enterprise applications, mobile, and Internet of Things (IoT) applications. In this chapter, you learn about different application-based vulnerabilities and related attacks such as privilege escalation, cross-site scripting, injection vulnerabilities, directory traversal, buffer overflows, race conditions, improper input handling, replay attacks, application programming interface (API) attacks, and many others.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities " section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 3-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 3-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Privilege Escalation	1
Cross-Site Scripting (XSS) Attacks	2
Injection Attacks	3
Pointer/Object Dereference	4
Directory Traversal	5
Buffer Overflows	6
Race Conditions	7
Error Handling	8
Improper Input Handling	9
Replay Attacks	10–11
Request Forgeries	12
Application Programming Interface (API) Attacks	13
Resource Exhaustion	14
Memory Leaks	15
Secure Socket Layer (SSL) Stripping	16
Driver Manipulation	17
Pass the Hash	18

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

---

1. What type of privilege escalation attack occurs when a user accesses functions or content reserved for other normal users—for example, if one user reads another's email?

a. Horizontal

b. Vertical

c. Sudo abuse

d. None of these answers are correct.

2. Which vulnerability can lead to stealing of cookies or redirecting users to malicious sites and where the malicious code or script is permanently stored on a vulnerable system?

a. DOM-based XSS

b. Stored XSS

c. Reflected XSS

d. All of these answers are correct.

3. Which type of injection attack occurs when code is run within the address space of another process, forcing it to load another library?

a. DLL injection

b. LDAP injection

c. SQL injection

d. None of these answers are correct.

4. Which condition occurs when a program dereferences a pointer that it expects to be valid but is null, which can cause the application to exit or

the system to crash?

a. Address space layout randomization

b. Tunnel-trap

c. Format string

d. Null pointer dereferences

5. Which type of attack occurs when an attacker tries to escape the web root folder and access arbitrary files by using ../../ in an URL?

a. Directory traversal

b. Path traversal

c. Dot-dot-slash

d. All of these answers are correct.

6. What occurs when arithmetic operations attempt to create a numeric value that is too big for the available memory space?

a. Stack overruns

b. Integer overflows

c. Format strings

d. Heap underruns

7. Race conditions are also known as _____ attacks.

a. Heap overflows

b. Time-of-check (TOC) or time-of-use (TOU)

c. Stack overflows

d. Buffer overflows

8. _____code should be checked thoroughly so that a malicious user can't find out any additional information about the system.

a. Regression

b. Patched

c. Error exception handling

d. Remote code execution

9. Input validation vulnerabilities can be found by using which of the following techniques?

a. A python debugger

b. Fuzzing

c. Heap flags

d. All of these answers are correct.

10. Which type of attack occurs when an attacker might use a packet sniffer to intercept data and retransmit it later?

a. Overflow

b. DLL injection

c. XSS

d. Replay

11. A web session ID is typically represented in a cookie. Which type of attack occurs when an attacker could steal a valid user's session ID and reuse it to perform malicious transactions?

a. Session replay

b. Session fuzzing

c. SQL injection

d. All of these answers are correct.

12. _____ attacks leverage the trust that the application has in the targeted user. For example, the attacker could inherit the privileges of the user to perform an undesired action, such as stealing sensitive information, creating users, or downloading malware.

a. XSS

b. SQLi

c. XML injection

d. XSRF

13. Which of the following are general best practices to protect APIs?

a. Secure API services to provide only HTTPS endpoints with a strong version of TLS.

b. Validate parameters in the application and sanitize incoming data from API clients.

c. Explicitly scan for common attack signatures; injection attacks often betray themselves by following common patterns.

d. All of these answers are correct.

14. Which of the following is a type of denial-of-service (DoS) attack?

a. C2

b. Botnet

c. Resource exhaustion

d. None of these answers are correct.

15. Which of the following attacks can reduce the performance of a computer, especially in systems with shared memory or limited memory?

a. Heap underrun

b. Format string

c. XSS

d. Memory leak

16. Which of the following is a common way to launch an SSL strip attack?

a. Creating a rogue wireless access point (hotspot)

b. Launching Metasploit

c. Fuzzing an application

d. Leveraging a memory leak

17. Which of the following is a driver manipulation attack where an attacker adds a small library that intercepts API calls?

a. Kernel module attack

b. Request forgery

c. Refactoring

d. Shimming

18. Pass the hash are attacks usually performed in which of the following operating systems?

a. Apple iOS

b. macOS

c. Linux

d. Windows

Foundation Topics

Privilege Escalation

Privilege escalation is the act of exploiting a bug or design flaw in a software or firmware application to gain access to resources that normally would have been protected from an application or user. This results in a user gaining additional privileges, more than were originally intended by the developer of the application—for example, if a regular user gains administrative control, or if a particular user can read another user's email without authorization.

The original developer does not intend for the attacker to gain higher levels of access but probably doesn't enforce a need-to-know policy properly and/or hasn't validated the code of the application appropriately. This technique is used by attackers to gain access to protected areas of operating systems or to applications—for example, if a particular user can read another user's email without authorization. Buffer overflows are used on Windows computers to elevate privileges as well. To bypass digital rights management (DRM) on games and music, attackers use a method known as jailbreaking, another type of privilege escalation, most commonly found on mobile devices. Malware also attempts to exploit privilege escalation vulnerabilities, if any exist on the system. Privilege escalation can also be attempted on network devices.

Generally, the fix for this is simply to update the device and to check on a regular basis if any updates are available. For example, a typical SOHO router has a user account and an administrator account. If a device like this isn't properly updated, an attacker can take advantage of a bug in the firmware to elevate the privileges of the user account. Couple this with the fact that a person forgot to put a password on the user account (or disable it) and your network could be in for some "fun." It is also possible on some devices to encrypt the firmware component. The following are a couple different types of privilege escalation:

- **Vertical privilege escalation:** When a lower-privileged user accesses functions reserved for higher-privileged users—for example, if a standard user can access functions of an administrator. This is also known as privilege elevation and is the most common description. To protect against this situation, you should update the network device firmware. In the case of an operating system, it should again be updated, and use of some type of access control system is also advisable—for example, User Account Control (UAC).
- **Horizontal privilege escalation:** When a normal user accesses functions or content reserved for other normal users—for example, if one user reads another's email. This can be done through hacking or by a person walking over to other people's computers and simply reading their email! Always have your users lock their computer (or log off) when they are not physically at their desk!

There is also privilege de-escalation, when high-privileged but segregated users can downgrade their access level to access normal users' functions. Sneaky administrators can attempt this to glean confidential information from an organization. It's a two-way street when it comes to security; you should think three-dimensionally when securing your network!

Cross-Site Scripting (XSS) Attacks

Cross-site scripting (commonly known as XSS) vulnerabilities, which have become some of the most common web application vulnerabilities, are

achieved using the following attack types:

- **Reflected XSS:** Reflected XSS attacks (nonpersistent XSS) occur when malicious code or scripts are injected by a vulnerable web application using any method that yields a response as part of a valid HTTP request. An example of a reflected XSS attack is a user being persuaded to follow a malicious link to a vulnerable server that injects (reflects) the malicious code back to the user's browser. This causes the browser to execute the code or script. In this case, the vulnerable server is usually a known or trusted site.
- **Stored (persistent) XSS:** Stored, or persistent, XSS attacks occur when the malicious code or script is permanently stored on a vulnerable or malicious server, using a database. These attacks are typically carried out on websites hosting blog posts (comment forms), web forums, and other permanent storage methods. An example of a stored XSS attack is a user requesting the stored information from the vulnerable or malicious server, which causes the injection of the requested malicious script into the victim's browser. In this type of attack, the vulnerable server is usually a known or trusted site.
- **DOM-based XSS:** The Document Object Model (DOM) is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure. DOM-based attacks are typically reflected XSS attacks that are triggered by sending a link with inputs that are reflected to the web browser. In DOM-based XSS attacks, the payload is never sent to the server. Instead, the payload is only processed by the web client (browser). In a DOM-based XSS attack, the attacker sends a malicious URL to the victim; and after the victim clicks on the link, it may load a malicious website or a site that has a vulnerable DOM route handler. After the vulnerable site is rendered by the browser, the payload executes the attack in the user's context on that site. One of the effects of any type of XSS attack is that the victim typically does not realize that an attack has taken place. DOM-based applications use global variables to manage client-side information. Often developers create unsecured applications

that put sensitive information in the DOM (for example, tokens, public profile URLs, private URLs for information access, cross-domain OAuth values, and even user credentials as variables). It is a best practice to avoid storing any sensitive information in the DOM when building web applications.

Successful exploitation of an XSS vulnerability could result in installation or execution of malicious code, account compromise, session cookie hijacking, revelation or modification of local files, or site redirection.

You typically find XSS vulnerabilities in the following:

- Search fields that echo a search string back to the user
- HTTP headers
- Input fields that echo user data
- Error messages that return user-supplied text
- Hidden fields that may include user input data
- Applications (or websites) that display user-supplied data

The following XSS test can be performed from a browser's address bar:

```
javascript:alert("Omar_s_XSS test");
javascript:alert(document.cookie);
```

The following XSS test can be performed in a user input field in a web form:

```
<script>alert("XSS Test")</script>
```

Attackers can use obfuscation techniques in XSS attacks by encoding tags or malicious portions of the script using Unicode so that the link or HTML content is disguised to the end user browsing the site.

Tip

The Open Web Application Security Project (OWASP) is a nonprofit organization where numerous security professionals contribute to projects designed to improve the security of web applications, as well as mobile and IoT devices. I strongly suggest leveraging the numerous resources and projects that

OWASP provides—not only for the exam but also to further your knowledge about application-based vulnerabilities, mitigations, and how to prevent them. OWASP provides a list of the top 10 vulnerabilities in web applications. You can find this list at https://owasp.org/www-project-top-ten. XSS has been on that list for many years.

OWASP also provides several recommendations and guidelines to help you prevent XSS at https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.

Injection Attacks

Attackers can use several types of ***injection attacks*** to manipulate and compromise an application. The following are the most prevalent injection attacks:

- Structured Query Language (SQL) injection attacks
- Dynamic link library (DLL) injection attacks
- Lightweight Directory Access Protocol (LDAP) injection attacks
- Extensible Markup Language (XML) injection attacks

Structured Query Language (SQL) Injection Attacks

SQL injection (SQLi) vulnerabilities can be catastrophic because they can allow an attacker to view, insert, delete, or modify records in a database. In an SQL injection attack, the attacker inserts, or injects, partial or complete SQL queries via the web application. The attacker injects SQL commands into input fields in an application or a URL in order to execute predefined SQL commands.

Let’s take a brief look at SQL. As you might know, the following are some of the most common SQL statements (commands):

- **SELECT:** Used to obtain data from a database
- **UPDATE:** Used to update data in a database
- **DELETE:** Used to delete data from a database
- **INSERT INTO:** Used to insert new data into a database
- **CREATE DATABASE:** Used to create a new database
- **ALTER DATABASE:** Used to modify a database
- **CREATE TABLE:** Used to create a new table
- **ALTER TABLE:** Used to modify a table
- **DROP TABLE:** Used to delete a table
- **CREATE INDEX:** Used to create an index or a search key element
- **DROP INDEX:** Used to delete an index

Typically, SQL statements and divided into the following categories:

- Data definition language (DDL) statements
- Data manipulation language (DML) statements
- Transaction control statements
- Session control statements
- System control statements
- Embedded SQL statements

Tip

At the W3Schools website, you can find a tool called the Try-SQL Editor that enables you to practice SQL statements in an “online database” (see https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all). You can use this tool to become familiar with SQL statements and how they may be passed to an

application. Another good online resource that explains SQL queries in detail is https://www.geeksforgeeks.org/sql-ddl-dml-tcl-dcl.

Web applications construct SQL statements involving SQL syntax invoked by the application mixed with user-supplied data. The first portion of the SQL statement shown in Figure 3-1 is not shown to the user; typically, the application sends this portion to the database behind the scenes. The second portion of the SQL statement is typically user input in a web form.

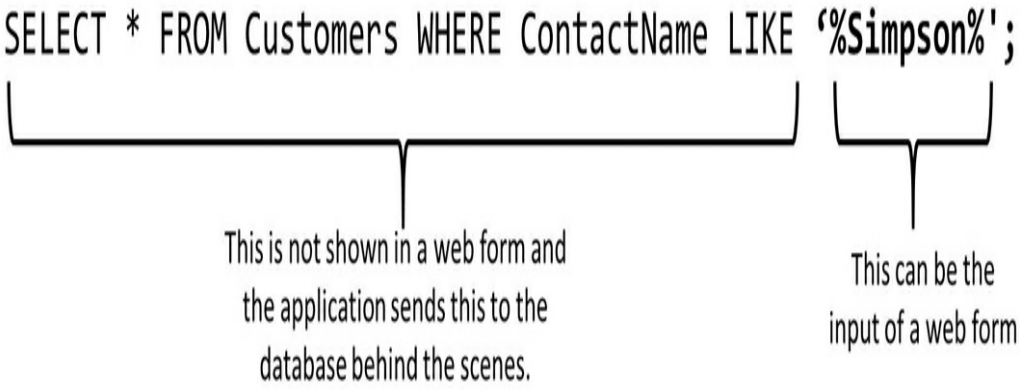


Figure 3-1 A SQL Query

If an application does not sanitize user input, an attacker can supply crafted input while trying to make the original SQL statement execute further actions in the database. SQL injections can be done using user-supplied strings or numeric input.

Tip

I have created a learning environment called WebSploit that you can use to practice some of these attacks in a safe environment. It includes several intentionally vulnerable applications running in Docker containers. Instructions on how to set up WebSploit can be found at https://websploit.org.

Figure 3-2 shows a basic SQL injection attack.

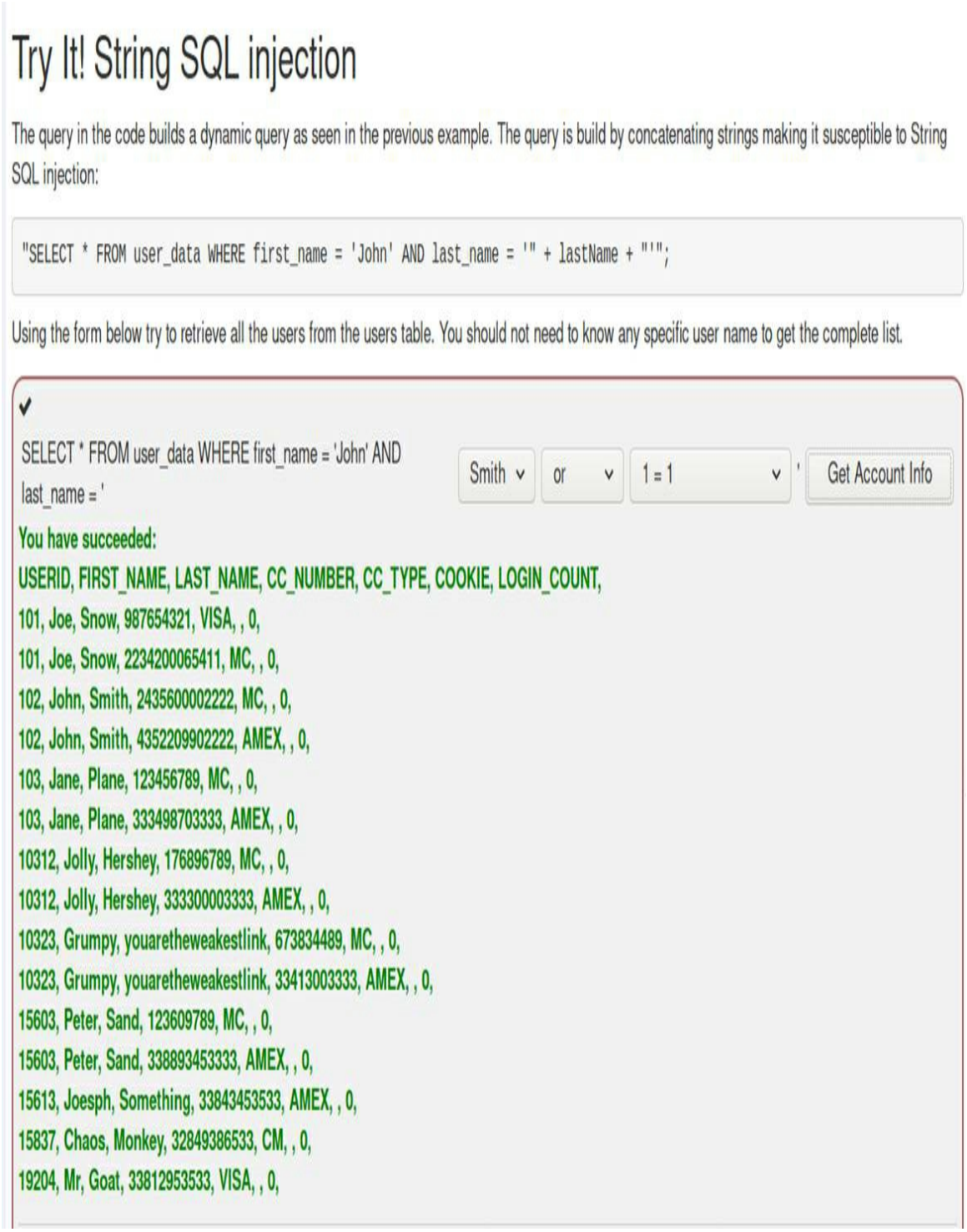


Figure 3-2 A Basic SQL Injection Attack

In the figure, WebGoat is used to demonstrate the effects of an SQL injection

attack. When the string **Smith' or '1'='1** is entered in the web form, it causes the application to display all records in the database table to the attacker.

One of the first steps when finding SQL injection vulnerabilities is to understand when the application interacts with a database. This is typically done with web authentication forms, search engines, and interactive sites such as e-commerce sites.

You can make a list of all input fields whose values could be used in crafting a valid SQL query. This includes trying to identify and manipulate hidden fields of POST requests and then testing them separately, plus trying to interfere with the query and to generate an error. As part of penetration testing, you should pay attention to HTTP headers and cookies.

As a penetration tester, you can start by adding a single quote (') or a semicolon (;) to the field or parameter in a web form. The single quote is used in SQL as a string terminator. If the application does not filter it correctly, you might be able to retrieve records or additional information that can help enhance your query or statement.

You can also use comment delimiters (such as -- or /* */), as well as other SQL keywords, including AND and OR operands. Another simple test is to insert a string where a number is expected.

SQL Injection Categories

SQL injection attacks can be divided into the following categories:

- **In-band SQL injection:** With this type of injection, the attacker obtains the data by using the same channel that is used to inject the SQL code. This is the most basic form of an SQL injection attack, where the data is dumped directly in a web application (or web page).
- **Out-of-band SQL injection:** With this type of injection, the attacker retrieves data using a different channel. For example, an email, a text, or an instant message could be sent to the attacker with the results of the query; or the attacker might be able to send the compromised data to

another system.

- **Blind (or inferential) SQL injection:** With this type of injection, the attacker does not make the application display or transfer any data; rather, the attacker is able to reconstruct the information by sending specific statements and discerning the behavior of the application and database.

Tip

To perform an SQL injection attack, an attacker must craft a syntactically correct SQL statement (query). The attacker may also take advantage of error messages coming back from the application and might be able to reconstruct the logic of the original query to understand how to execute the attack correctly. If the application hides the error details, the attacker might need to reverse-engineer the logic of the original query.

Essentially, five techniques can be used to exploit SQL injection vulnerabilities:

- **Union operator:** This technique is typically used when a SQL injection vulnerability allows a SELECT statement to combine two queries into a single result or a set of results.
- **Boolean:** This technique is used to verify whether certain conditions are true or false.
- **Error-based technique:** This technique is used to force the database to generate an error in order to enhance and refine an attack (injection).
- **Out-of-band technique:** This technique is typically used to obtain records from the database by using a different channel. For example, it is possible to make an HTTP connection to send the results to a different web server or a local machine running a web service.
- **Time delay:** It is possible to use database commands to delay answers. An attacker may use this technique when he or she doesn’t get any output or error messages from the application.

It is possible to combine any of the techniques described here to exploit an SQL injection vulnerability. For example, an attacker may use the union operator and out-of-band techniques.

Dynamic Link Library (DLL) Injection Attacks

In ***DLL injection***, code is run within the address space of another process by forcing it to load a dynamic link library (DLL). Ultimately, this type of attack can influence the behavior of a program that was not originally intended. It can be uncovered through penetration testing.

Binary planting is another term used when the attacker implants a malicious binary to a local or remote file system in order for the application to load and execute such malicious binary. You can access detailed information about this technique at OWASP's website at https://owasp.org/www-community/attacks/Binary_planting.

Lightweight Directory Access Protocol (LDAP) Injection Attacks

LDAP injection is similar to SQL injection, again using a web form input box to gain access, or by exploiting weak LDAP lookup configurations. The Lightweight Directory Access Protocol is used to maintain a directory of information such as user accounts or other types of objects. The best way to protect against this type of attack (and all code-injection techniques for that matter) is to incorporate strong input validation.

Extensible Markup Language (XML) Injection Attacks

XML injection attacks can compromise the logic of Extensible Markup Language (XML) applications—for example, XML structures that contain the code for users. It can be used to create new users and possibly obtain administrative access. You can test for this type of attack by attempting to insert XML metacharacters such as single and double quotes. It can be prevented by filtering *in* allowed characters (for example, A–Z only). This is an example of "default deny" where only what you explicitly filter in is permitted; everything else is forbidden.

One thing to remember is that when attackers utilize code-injecting techniques, they are adding their own code to existing code or are inserting their own code into a form. A variant of this is command injection, which doesn't utilize new code; instead, an attacker executes system-level commands on a vulnerable application or OS. The attacker might enter the command (and other syntax) into an HTML form field or other web-based form to gain access to a web server's password files.

XML External Entity (XXE) is another type of input validation vulnerability and attack against an application that parses XML input. It occurs when input containing a reference to an XML external entity is handled by a poorly configured XML parser. Successful exploitation could lead to the disclosure of confidential data, server-side request forgery, and denial of service. To obtain additional information about this attack and vulnerability, see https://owasp.org/www-community/attacks/.

Pointer/Object Dereference

Another potential memory-related issue deals with pointer dereferencing—for example, the null pointer dereference. Pointer dereferencing is common in programming; when you want to access data (say, an integer) in memory, dereferencing the pointer would retrieve different data from a different section of memory (perhaps a different integer). Programs that contain a ***null pointer dereference*** generate memory fault errors (memory leaks). A null

pointer dereference occurs when the program dereferences a pointer that it expects to be valid but is null, which can cause the application to exit or the system to crash. From a programmatical standpoint, the main way to prevent this situation is to use meticulous coding. Programmers can use special memory error analysis tools to enable error detection for a null pointer deference. Once the problem is identified, the programmer can correct the code that may be causing the error(s). But this concept can be used to attack systems over the network by initiating IP address to hostname resolutions—ones that the attacker hopes will fail—causing a return null. What this all means is that the network needs to be protected from attackers attempting this (and many other) programmatical and memory-based attacks via a network connection.

Note

For more information on null pointer dereferencing (and many other software weaknesses), see the Common Weakness Enumeration portion of MITRE at https://cwe.mitre.org/. Add that site to your favorites!

A programmer can make use of ***address space layout randomization (ASLR)*** to help prevent the exploitation of buffer overflows, remote code execution, and memory corruption vulnerabilities. It randomly arranges the different address spaces used by a program (or process). This can aid in protecting mobile devices (and other systems) from exploits caused by memory-management problems. While there are exploits to ASLR (such as side-channel attacks) that can bypass it and derandomize how the address space is arranged, many systems employ some version of ASLR.

Directory Traversal

Directory traversal, path traversal, or the ../ (“dot-dot-slash”) attack, is a method of accessing unauthorized parent (or worse, root) directories. It is

often used on web servers that have PHP files and are Linux or UNIX-based, but it can also be perpetrated on Microsoft operating systems (in which case it would be ..\ or the “dot-dot-backslash” attack). It is designed to get access to files such as ones that contain passwords. This access can be prevented by updating the operating system or by checking the code of files for vulnerabilities, otherwise known as fuzzing. For example, a PHP file on a Linux-based web server might have a vulnerable **if** or **include** statement, which when attacked properly could give the attacker access to higher directories and the passwd file.

Figure 3-3 shows an example of a directory traversal (path traversal) attack. The attacker is able to retrieve the contents of the /etc/passwd file of a vulnerable web application. You can also practice this attack using the WebSploit environment.

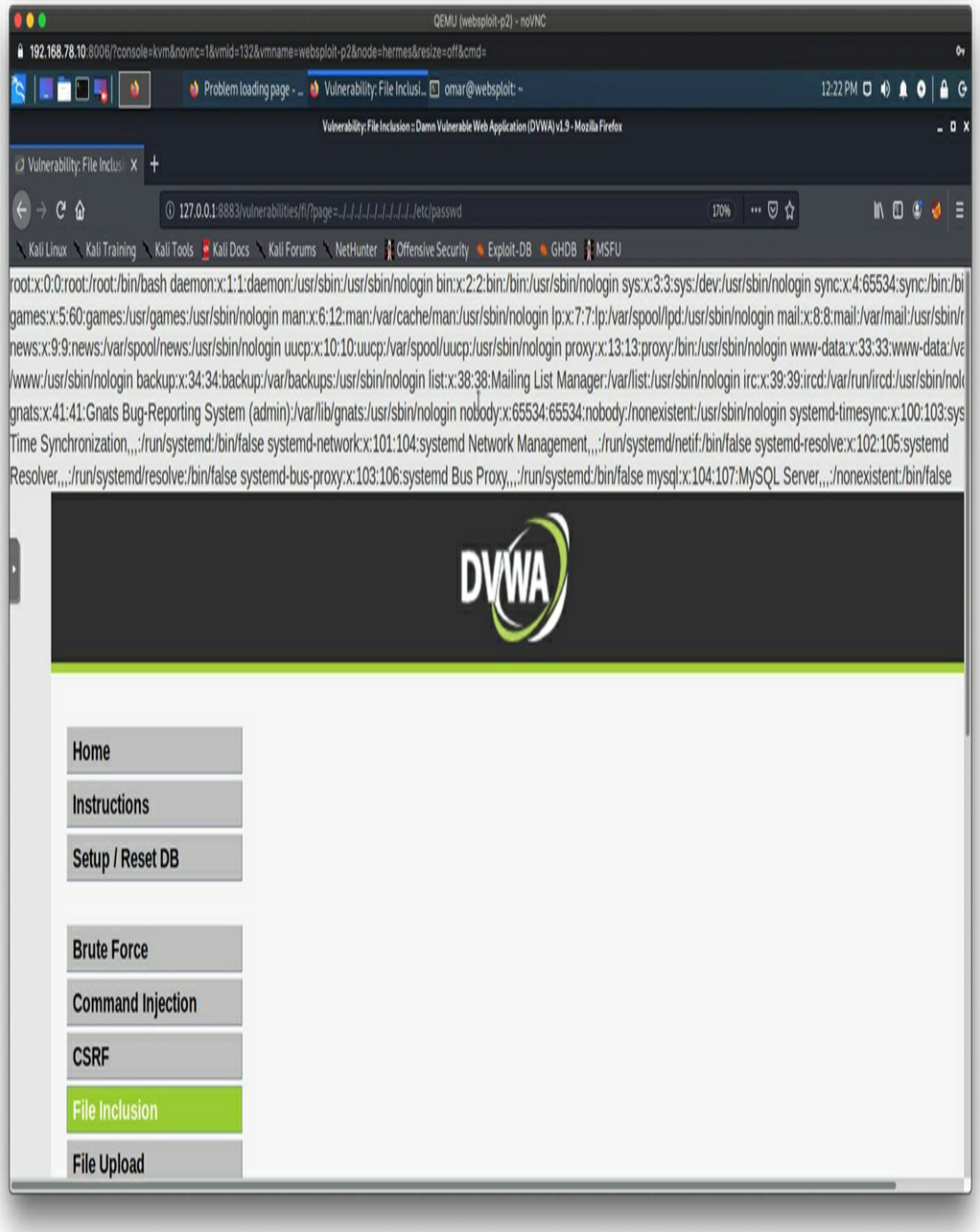


Figure 3-3 A Directory Traversal Attack

Buffer Overflows

Memory and buffer vulnerabilities are common. Of the several types, perhaps most important is the buffer overflow. A ***buffer overflow*** occurs when a process stores data outside the memory that the developer intended. This could cause erratic behavior in the application, especially if the memory already had other data in it. Stacks and heaps are data structures that can be affected by buffer overflows. The stack is a key data structure necessary for the exchange of data between procedures. The heap contains data items whose size can be altered during execution. Value types are stored in a stack, whereas reference types are stored in a heap. An ethical coder will try to keep these running efficiently. An unethical coder wanting to create a program vulnerability could, for example, omit input validation, which could allow a buffer overflow to affect heaps and stacks, which in turn could adversely affect the application or the operating system in question.

Let's say a programmer allows for 16 bytes in a string variable. This wouldn't be a problem normally; however, if the programmer failed to verify that *no more than* 16 bytes could be copied over to the variable, that would create a vulnerability that an attacker could exploit with a buffer overflow attack. The buffer overflow can also be initiated by certain inputs. For example, corrupting the stack with no-operation (no-op, NOP, or NOOP) machine instructions, which when used in large numbers can start a NOP slide, can ultimately lead to the execution of unwanted arbitrary code, or can lead to a denial of service on the affected computer.

All this can be prevented by patching the system or application in question, making sure that the OS uses data execution prevention, and utilizing bounds checking, which is a programmatic method of detecting whether a particular variable is within design bounds before it is allowed to be used. It can also be prevented by using correct code, checking code carefully, and using the right programming language for the job in question (the right tool for the right job, yes?). Without getting too much into the programming side of things, special values called "canaries" are used to protect against buffer overflows.

On a semi-related note, ***integer overflows*** occur when arithmetic operations attempt to create a numeric value that is too big for the available memory space. This creates a *wrap* and can cause resets and undefined behavior in programming languages such as C and C++. The security ramification is that the integer overflow can violate the program's default behavior and possibly lead to a buffer overflow. This situation can be prevented or avoided by making overflows trigger an exception condition or by using a model for automatically eliminating integer overflow, such as the CERT As-if Infinitely Ranged (AIR) integer model. You can learn more about this model at http://www.cert.org/secure-coding/tools/integral-security.cfm.

Then there are memory leaks. A ***memory leak*** is a type of resource leak caused when a program does not release memory properly. The lack of freed-up memory can reduce the performance of a computer, especially in systems with shared memory or limited memory. A kernel-level leak can lead to serious system stability issues. The memory leak might happen on its own due to poor programming, or it could be that code resides in the application that is vulnerable and is later exploited by an attacker who sends specific packets to the system over the network. This type of error is more common in languages such as C or C++ that have no automatic garbage collection, but it could happen in any programming language. Several memory debuggers can be used to check for leaks. However, it is recommended that garbage collection libraries be added to C, C++, or other programming language, to check for potential memory leaks.

Arbitrary Code Execution/Remote Code Execution

In arbitrary code execution an attacker obtains control of a target computer through some sort of vulnerability, thus gaining the power to execute commands on that remote computer at will. Programs that are designed to exploit software bugs or other vulnerabilities are often called arbitrary code execution exploits. These types of exploits inject "shellcode" to allow the attacker to run arbitrary commands on the remote computer. This type of attack is also known as ***remote code execution (RCE)*** and can potentially allow the attacker to take full control of the remote computer and turn it into

a zombie.

RCE commands can be sent to the target computer using the URL of a browser or by using the Netcat service, among other methods. To defend against this type of attack, you should update applications, or if the application is being developed by your organization, it should be checked with fuzz testing and strong input validation (client side and server side) as part of the software development lifecycle (SDLC).

Note

SDLC is covered in more detail in Chapter 11, "Summarizing Secure Application Development, Deployment, and Automation Concepts."

If you have PHP running on a web server, it can be set to disable remote execution of configurations. A web server (or other server) can also be configured to block access from specific hosts.

Note

RCE is also very common with web browsers. All browsers have been affected at some point, though some instances are more publicized than others. To see proof of this, access the Internet and search for the Common Vulnerabilities and Exposures (CVE) list for each type of web browser.

Race Conditions

Another exploit is the ***race condition***. This exploit is difficult to perform because it takes advantage of the small window of time between when a service is used and its corresponding security control is executed in an application or OS, or when temporary files are created. It can be defined as

anomalous behavior due to a dependence on timing of events. Race conditions are also known as ***time of check (TOC) or time of use (TOU)*** attacks. Imagine that you are tasked with changing the permissions to a folder or changing the rights in an ACL. If you remove all of the permissions and apply new permissions, then there will be a short period of time when the resource (and system) might be vulnerable. This vulnerability depends on the system used, how it defaults, and how well you have planned your security architecture. That was a basic example, but the race condition is more common within the programming of an application. This exploit can be prevented by proper secure coding of applications and planning of the system and network architecture.

Error Handling

At times, applications will fail. How they fail determines their security. Failure exceptions might show the programming language that was used to build the application, or worse, lead to access holes. ***Error handling*** or error exception handling code should be checked thoroughly so that a malicious user can't find out any additional information about the system. These error handling methods are sometimes referred to technically as pseudocodes. For example, to handle a program exception, properly written pseudocode will basically state (in spoken English): "If a program module crashes, then restart the program module."

Once found, security vulnerabilities should be thoroughly tested, documented, and understood. Patches should be developed to fix the problem but not cause other issues or application regression.

There are some other concepts to consider. First is *obfuscation*, which is the complicating of source code to make it more difficult for people to understand. This is done to conceal its purpose in order to prevent tampering and/or reverse engineering. It is an example of security through obscurity. Other examples of security through obscurity include code camouflaging and steganography, both of which might be used to hide the true code being used. Another important concept is *code checking*, which involves limiting the

reuse of code to that which has been approved for use and removing dead code. It's also vital to incorporate good memory management techniques. Finally, you should be very careful when using third-party libraries and software development kits (SDKs) and test them thoroughly before using them within a live application.

For the Security+ exam, the most important of the SDLC phases are maintenance and testing. In the maintenance phase, which doesn't end until the software is removed from all computers, an application needs to be updated accordingly, corrected when it fails, and constantly monitored. It's imperative that you know some of the vulnerabilities and attacks to a system or application, and how to fix them and protect against them. The best way to prevent these attacks is to test and review code.

Improper Input Handling

Programmers have various ways to test their code, including system testing, ***input validation***, and fuzzing. When a combination of these testing techniques is used during development and testing, there is a much higher probability of a secure application as the end result.

System testing is generally broken down into two categories: black-box and white-box. ***Black-box testing*** utilizes people who do not know the system. These people (or programs, if automated) test the functionality of the system. Specific knowledge of the system code—and programming knowledge in general—is not required. The tester does not know about the system's internal structure and is often given limited information about what the application or system is supposed to do. In black-box testing, one of the most common goals is to crash the program (often done using fuzzing tools or "fuzzers"). If a user is able to crash the program by entering improper input, the programmer has probably neglected to thoroughly check the error handling code and/or input validation. ***White-box testing*** (also known as transparent testing) is a way of testing the internal workings of the application or system. Testers must have programming knowledge and are given detailed information about the design of the system. They are given

login details, production documentation, and source code. System testers might use a combination of fuzzing (covered shortly), data flow testing, and other techniques such as stress testing, penetration testing, and sandboxes.

Note

A third category that has become popular is gray-box testing, where the tester has internal knowledge of the system from which to create tests but conducts the tests the way a black-box tester would—at the user level.

Stress testing is usually done on real-time operating systems, mission-critical systems, and software, and checks if they have the robustness and availability required by the organization. In a penetration test, the tester mimics what an attacker could do in a vulnerable system.

Compile-Time Errors vs. Runtime Errors

Programmers and developers need to test for potential compile-time errors and runtime errors. Compile time refers to the duration of time during which the statements written in any programming language are checked for errors. ***Compile-time errors*** might include syntax errors in the code and type-checking errors. A programmer can check these without actually "running" the program but instead check it in the compile stage when it is converted into machine code.

A ***runtime error*** is a program error that occurs while the program is running. The term is often used in contrast to other types of program errors, such as syntax errors and compile-time errors. Runtime errors might include running out of memory, invalid memory address access, invalid parameter value, or buffer overflows/dereferencing a null pointer (to name a few), all of which can be discovered only by running the program as a user. Another potential runtime error can occur if there is an attempt to divide by zero. These types of errors result in a software exception. Software and hardware exceptions need to be handled properly. Consequently, ***structured exception handling (SEH)*** is a mechanism used to handle both types of exceptions. It enables the programmer to have complete control over how exceptions are handled and

provides support for debugging.

Code issues and errors that occur in either compile time or run time could lead to vulnerabilities in the software. However, it's the runtime environment that you are more interested in from a security perspective, because that more often is where the attacker will attempt to exploit software and websites.

Input validation is very important for website design and application development. ***Input validation***, or data validation, is a process that ensures the correct usage of data: it checks the data that is inputted by users into web forms and other similar web elements. Data not being validated correctly can lead to various security vulnerabilities, including sensitive data exposure and the possibility of data corruption. You can validate data in many ways, from coded data checks and consistency checks to spelling and grammar checks, and so on. Whatever data is being dealt with, it should be checked to make sure it is being entered correctly and won't create or take advantage of a security flaw. If validated properly, bad data and malformed data will be rejected. Input validation should be done both on the client side and, more importantly, on the server side. Let's look at an example next.

If an organization has a web page with a PHP-based contact form, the data entered by the visitor should be checked for errors or maliciously typed input. The following PHP code is contained within a common contact form:

```
else if (!preg_match('/^[A-Za-z0-9.-]+$/', $domain))
 {
 // character not valid in domain part
 $isValid = false;
 }
```

This example is a part of a larger piece of code that checks the entire email address a user enters into a form field. This particular snippet of code checks to make sure the user is not trying to enter a backslash in the domain name portion of the email address. This character is not allowed in email addresses and could be detrimental if used maliciously. In the first line within the brackets, note that it says A-Za-z0-9.-, which tells the system what characters are allowed. Uppercase and lowercase letters, numbers, periods, and dashes

are allowed, but other characters such as backslashes, dollar signs, and so on are not allowed. The form's supporting PHP files would interpret those other characters as illegitimate data and would not pass them on through the system. The user would receive an error, which is a part of client-side validation. But the more a PHP form is programmed to check for errors, the more it is possible to have additional security holes. Therefore, server-side validation is even more important. Any data that *is* passed on by the PHP form should be checked at the server as well. In fact, an attacker might not even be using the form in question but instead might be attacking the URL of the web page in some other manner. This type of validation can be checked at the server within the database software or through other means. By the way, the concept of combining client-side and server-side validation also goes for pages that utilize JavaScript.

This is just one basic example, but as mentioned previously, input validation is the key to preventing attacks such as SQL injection and XSS. All form fields should be tested for good input validation code, both on the client side and the server side. By combining the two and checking every access attempt, you develop complete mediation of requests.

Note

Using input validation is one way to prevent *sensitive data exposure*, which occurs when an application does not adequately protect personally identifiable information (PII). You can also prevent this type of exposure by doing the following: making sure that inputted data is never stored or transmitted in clear text; using strong encryption and securing key generation and storage; using HTTPS for authentication; and using a salt, which is random data used to strengthen hashed passwords.

Replay Attacks

A ***replay attack*** is a network attack in which a valid data transmission is

maliciously or fraudulently repeated or delayed. It differs from session hijacking in that the original session is simply intercepted and analyzed for later use. In a replay attack, an attacker might use a packet sniffer to intercept data and retransmit it later. In this way, the attacker can impersonate the entity that originally sent the data. For example, if customers were to log in to a banking website with their credentials while an attacker was watching, the attacker could possibly sniff out the packets that include the usernames and passwords and then possibly connect with those credentials later on. Of course, if the bank uses Secure Socket Layer (SSL) or Transport Layer Security (TLS) to secure login sessions, then the attacker would have to decrypt the data as well, which could prove more difficult. An organization can defend against this attack in several ways. The first is to use session tokens that are transmitted to people the first time they attempt to connect and identify them subsequently. They are handed out randomly so that attackers cannot guess at token numbers. The second way is to implement timestamping and synchronization, as in a Kerberos environment. A third way would be to use a timestamped ***nonce***, a random number issued by an authentication protocol that can be used only one time. You can also implement CHAP-based authentication protocols to provide protection against replay attacks.

Note

A replay attack should not be confused with SMTP relay, which occurs when one server forwards email to other email servers.

Session replay attacks occur when an attacker steals a user's valid session ID and reuses that ID to perform malicious transactions and activities with a web application.

Session hijacking is the exploitation of a computer session in an attempt to gain unauthorized access to data, services, or other resources on a computer. A few types of session hijacks can occur:

- **Session theft:** This hijack can be accomplished by making use of packet

header manipulation or by stealing a cookie from the client computer, which authenticates the client computer to a server. This is done at the application layer, and the cookies involved are often based off their corresponding web applications (such as WWW sessions). This type of attack can be combated by using encryption and long random numbers for the session key and then regeneration of the session after a successful login. The Challenge Handshake Authentication Protocol (CHAP) can also be employed to require clients to periodically reauthenticate. However, session hijacking can also occur at the network layer—for example, TCP/IP hijacking.

Note

Details about different authentication protocols are covered in Chapter 24, "Implementing Authentication and Authorization Solutions."

- **TCP/IP hijacking:** This is a common type of session hijacking, due to its popularity among attackers. It occurs when an attacker takes over a TCP session between two computers without the need of a cookie or any other type of host access. Because most communications' authentication occurs only at the beginning of a standard TCP session, an attacker can attempt to gain access to a client computer anytime after the session begins. One way would be to spoof the client computer's IP address, then find out what was the last packet sequence number sent to the server, and then inject data into the session before the client sends another packet of information to the server. Remember the three-way handshake that occurs at the beginning of a session; this is the only authentication that occurs during the session. A synchronization (SYN) packet is sent by the client to the server, then a SYN/ACK packet is sent by the server to the client, and finally, an acknowledgment (ACK) packet is sent by the client to the server. An attacker can jump in anytime after this process and attempt to steal the session by injecting data into the data stream. This is the more difficult part: the attacker might need to perform a DoS attack on the client to stop it from sending any more packets so that the packet sequence number doesn't increase.

In contrast, UDP sessions are easier to hijack because no packet sequence numbers exist. Targets for this type of attack include online games and also DNS queries. To mitigate the risk of TCP/IP hijacking, you should employ encrypted transport protocols such as SSL, IPsec, and SSH.

- **Blind hijacking:** This type of hijacking occurs when an attacker blindly injects data into a data stream without being able to see whether the injection was successful. The attacker could be attempting to create a new administrator account or gain access to one.
- **Clickjacking:** This type of hijacking occurs when a user browsing the web is tricked into clicking something different from what the user thought he or she was clicking. It is usually implemented as a concealed link—embedded code or a script on a website that executes when the user clicks that element. For example, a Flash script, when clicked, could cause the user's webcam to turn on without the user's consent. The user is often redirected to the website from a malicious source. This type of attack can be prevented by updating the user's web browser and using third-party add-ons that watch for clickjacking code or scripts. On the server side, web page frames (such as iframes) must be managed carefully. JavaScript-based snippets can be added and content security policies also can be configured to help manage frames.
- **On-path attack (previously known as man-in-the-middle [MITM] and man-in-the-browser [MITB] attacks):** These attacks intercept all data between a client and a server. It is a type of active interception. If successful, all communications now go through the MITM attacking computer. The attacking computer can at this point modify the data, insert code, and send it to the receiving computer. This type of eavesdropping is successful only when the attacker can properly impersonate each endpoint. Cryptographic protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS) address MITM attacks by using a mutually trusted third-party certification authority (CA). These public key infrastructures (PKIs) should use strong mutual authentication such as secret keys and strong passwords.

On-path attacks also can make use of a Trojan (from a proxy location) that infects a vulnerable web browser and modifies web pages and online

transactions, in an attempt to ultimately steal money or data. For example, a user might make an online banking transaction, and the user would see confirmation of the exact transaction, but on the banking side, a different amount might have been actually transferred, with some of it going to a different location altogether. This type of attack can be prevented by updating the web browser, using transaction verification (often third-party), and updating the antimalware on the computer in question.

- ***Watering hole attack*:** As discussed in Chapter 1, a watering hole attack is a targeted attack that occurs when an attacker profiles the websites that the intended victim accesses. The attacker then scans those websites for possible vulnerabilities. If the attacker locates a website that can be compromised, the website is then injected with a JavaScript or other similar code injection that is designed to redirect the user when the user returns to that site (also known as a pivot attack). The user is then redirected to a site with some sort of exploit code…and the rest is, well, history. The purpose is to infect computers in the organization's network, thereby allowing the attacker to gain a foothold in the network for espionage or other reasons. Watering hole attacks are often designed to profile users of specific organizations, and as such, an organization should develop policies to prevent these attacks. This can be done by updating antimalware applications regularly and by using secure virtual browsers that have little connectivity to the rest of the system and the rest of the network. To avoid having a website compromised as part of this attack, you, as administrator, should use proper programming methods and scan the website for malware regularly.

Request Forgeries

Cross-site request forgery (XSRF) is a type of vulnerability in which an attacker lures the targeted user to execute unwanted actions on a web application. Threat-performing XSRF attacks leverage the trust that the application has in the targeted user. For example, the attacker could inherit the privileges of the user to perform an undesired action, such as stealing sensitive information, creating users, or downloading malware.

A ***server-side request forgery (SSRF)***, unlike a CSRF, is initiated from a web server through a vulnerable web application. With a CSRF attack, the user is tricked into doing something that benefits the attacker. In contrast, an SSRF attack is done for the purpose of compromising information from the web server or enabling other attacks, such as bypassing input validation controls or enabling the attacker to execute further commands. As SSRF attack exploits trust relationships.

Application Programming Interface (API) Attacks

Application programming interfaces (APIs) are used everywhere today. A large number of modern applications use some type of API to allow other systems to interact with the application. Unfortunately, many APIs lack adequate controls and are difficult to monitor. The breadth and complexity of APIs also make it difficult to automate effective security testing. Attackers can launch attacks against APIs to steal, delete, or modify data or to perform a denial-of-service condition.

The following are a few methods or technologies behind modern APIs:

- **Simple Object Access Protocol (SOAP):** This standards-based web services access protocol was originally developed by Microsoft and has been used by numerous legacy applications for many years. SOAP exclusively uses XML to provide API services. XML-based specifications are governed by XML Schema Definition (XSD) documents. SOAP was originally created to replace older solutions such as the Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). You can find the latest SOAP specifications at https://www.w3.org/TR/soap.
- **Representational State Transfer (REST):** This API standard is easier to use than SOAP. It uses JSON instead of XML, and it uses standards such as Swagger and the OpenAPI Specification (https://www.openapis.org) for ease of documentation and to encourage adoption.

- **GraphQL:** GraphQL is a query language for APIs that provides many developer tools. GraphQL is now used for many mobile applications and online dashboards, and many different languages support GraphQL. You can learn more about it at https://graphql.org/code.

SOAP and REST use the HTTP protocol; however, SOAP limits itself to a stricter set of API messaging patterns than REST.

An API often provides a roadmap that describes the underlying implementation of an application. API documentation can provide a great level of detail that can be very valuable to a security professional, as well to attackers. API documentation can include the following:

- **Swagger (OpenAPI):** Swagger is a modern framework of API documentation and development that is the basis of the OpenAPI Specification (OAS). You can obtain additional information about Swagger at https://swagger.io. The OAS specification is available at https://github.com/OAI/OpenAPI-Specification.
- **Web Services Description Language (WSDL) documents:** WSDL is an XML-based language that is used to document the functionality of a web service. The WSDL specification can be accessed at https://www.w3.org/TR/wsdl20-primer.
- **Web Application Description Language (WADL) documents:** WADL is an XML-based language for describing web applications. The WADL specification can be obtained from https://www.w3.org/Submission/wadl.

The following are general best practices and recommendations for securing APIs:

- Secure API services to only provide HTTPS endpoints with a strong version of TLS.
- Validate parameters in the application and sanitize incoming data from API clients.
- Explicitly scan for common attack signatures; injection attacks often betray themselves by following common patterns.
- Use strong authentication and authorization standards.

- Use reputable and standard libraries to create the APIs.
- Segment API implementation and API security into distinct tiers; doing so frees up the API developer to focus completely on the application domain.
- Identify what data should be publicly available and what is sensitive information.
- If possible, have a security expert do the API code verification.
- Make internal API documentation mandatory.
- Avoid discussing company API development (or any other application development) on public forums.

Resource Exhaustion

Resource exhaustion is an attack against availability that is designed to bring the network, or access to a particular TCP/IP host/server, to its knees by flooding it with useless traffic. Resource exhaustion attacks are a form of denial-of-service attacks. They can also leverage software vulnerabilities such as memory leaks and file descriptor leaks.

Today, most DoS attacks are launched via botnets (a collection of compromised systems), whereas in the past tools such as the Ping of Death or Teardrop may have been used.

Memory Leaks

A ***memory leak*** is a type of resource leak caused when a program does not release memory properly. The lack of freed-up memory can reduce the performance of a computer, especially in systems with shared memory or limited memory. A kernel-level leak can lead to serious system stability issues. The memory leak might happen on its own due to poor programming,

or it could be that code resides in the application that is vulnerable and is later exploited by an attacker who sends specific packets to the system over the network. This type of error is more common in languages such as C or C++ that have no automatic garbage collection, but it could happen in any programming language. Several memory debuggers can be used to check for leaks. However, it is recommended that garbage collection libraries be added to C, C++, or other programming language to check for potential memory leaks.

Secure Socket Layer (SSL) Stripping

Transport Layer Security is the protocol that replaced SSL. However, many people still refer to TLS implementations as "SSL." TLS versions older than 1.2 and all SSL versions are susceptible to SSL stripping attacks. The latest version of TLS (at the time of writing) is version 1.3.

An ***SSL stripping attack*** occurs when an attacker performs a man-in-the-middle attack and can redirect a client to an insecure HTTP connection. The attacker still establishes a secure HTTPS connection between herself or himself and the server, as illustrated in Figure 3-4.

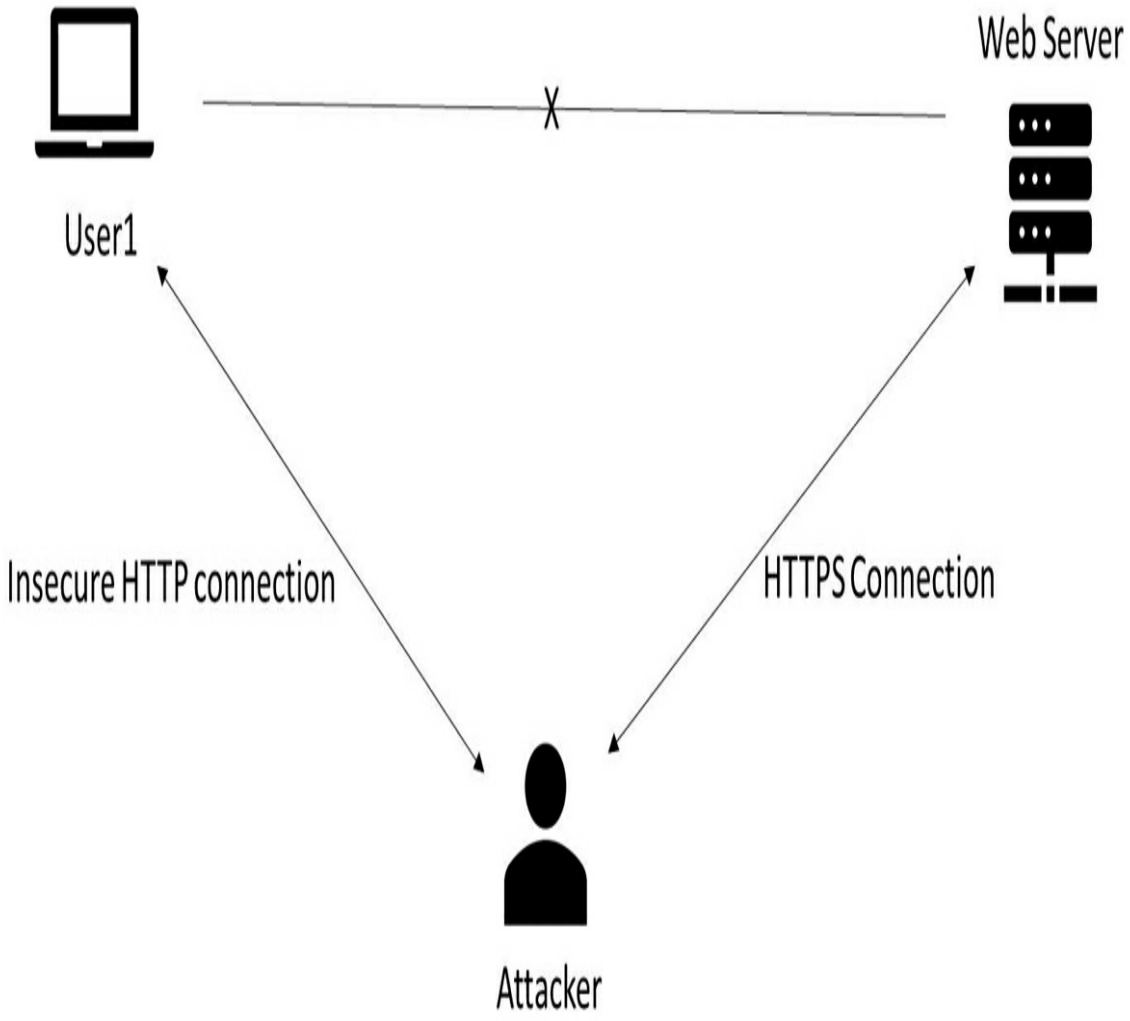


Figure 3-4 An SSL Stripping Attack

The attacker in Figure 3-4 downgrades the client connection from HTTPS to HTTP and sends it back to the victim's (User1) browser. The browser is then redirected to the attacker, and all the victim's data will be transferred in plaintext format. Meanwhile, the web server successfully established a secure connection, but with the attacker's machine, not the victim's system.

An attacker can launch an SSL strip attack in many different ways. One of the most common ways is to create a wireless hotspot and lure the victims to connect to it.

Driver Manipulation

It's important to verify that any drivers installed will be compatible with a system. As a security administrator, you should be aware of the potential to modify drivers through the use of driver ***shimming*** (the adding of a small library that intercepts API calls) and driver ***refactoring*** (the restructuring of driver code). By default, shims are not supposed to be used to resolve compatibility issues with device drivers, but it's impossible to foresee the types of malicious code that may present itself in the future. So, you should be careful when updating drivers by making sure that the drivers are signed properly and first testing them on a closed system.

Pass the Hash

All versions of Windows store passwords as hashes in a file called the Security Accounts Manager (SAM) file. The operating system does not know what the actual password is because it stores only a hash of the password. Instead of using a well-known hashing algorithm, Microsoft created its own implementation that has developed over the years.

Microsoft also has a suite of security protocols for authentication, called NT LAN Manager (NTLM). NTLM had two versions: NTLMv1 and NTLMv2. Since Windows 2000, Microsoft has used Kerberos in Windows domains. However, NTLM may still be used when the client is authenticating to a server via IP address or if a client is authenticating to a server in a different Active Directory (AD) forest configured for NTLM trust instead of a transitive interforest trust. In addition, NTLM might also still be used if the client is authenticating to a server that doesn't belong to a domain or if the Kerberos communication is blocked by a firewall.

So, what is a ***pass the hash attack***? Because password hashes cannot be reversed, instead of trying to figure out what the user's password is, an

attacker can just use a password hash collected from a compromised system and then use the same hash to log in to another client or server system.

Figure 3-5 shows an example of a pass the hash attack.

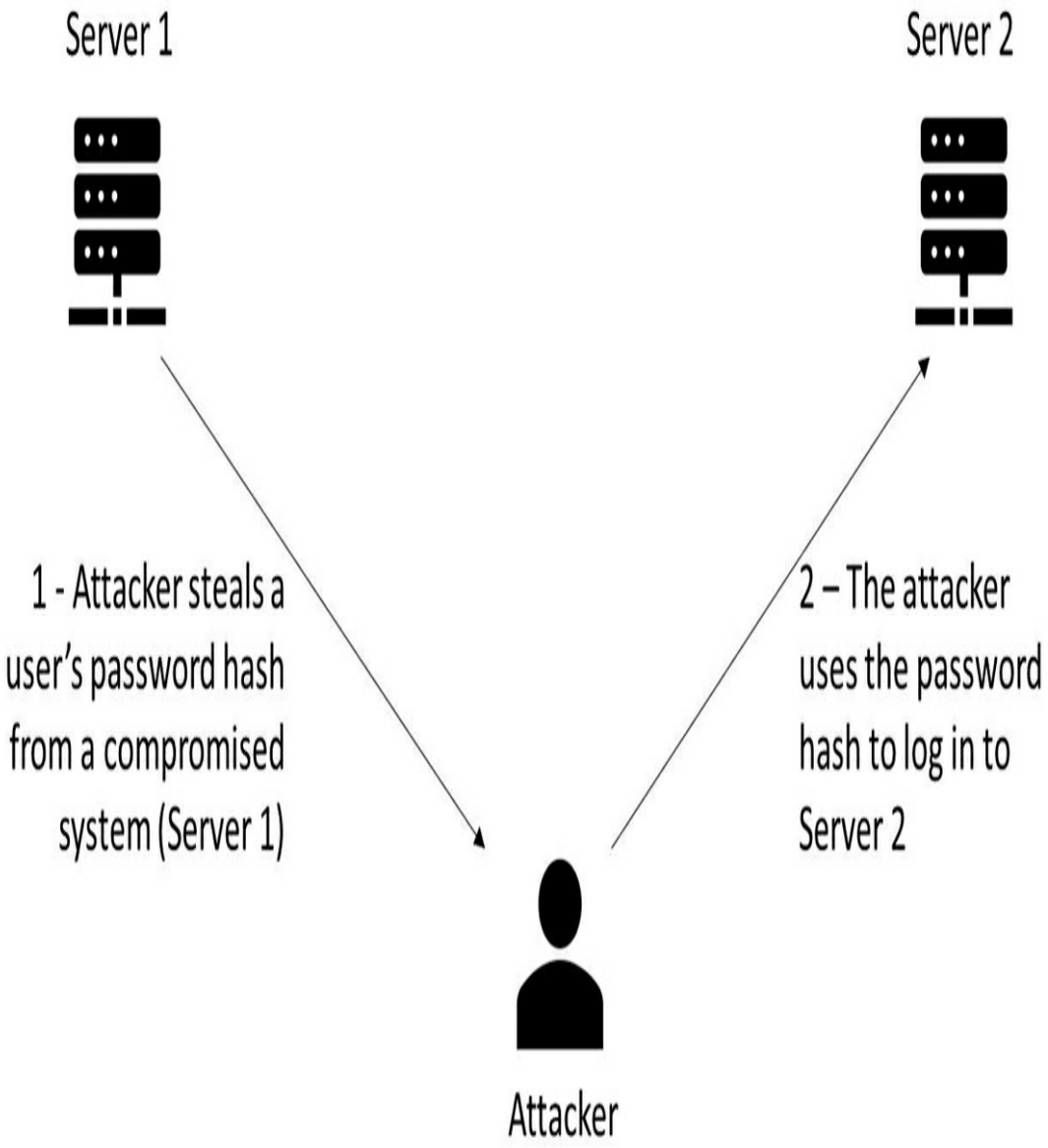


Figure 3-5 The Pass the Hash Attack

The Windows operating system and Windows applications ask users to enter their passwords when they log in. The system then converts the passwords into hashes (in most cases using an API called LsaLogonUser). A pass the hash attack goes around this process and just sends the hash to the system to authenticate.

Mimikatz is a tool used by many penetration testers, attackers, and even malware that can be useful for retrieving password hashes from memory; it is a very useful postexploitation tool. You can download the Mimikatz tool from https://github.com/gentilkiwi/mimikatz. Metasploit penetration testing software also includes Mimikatz as a Meterpreter script to facilitate exploitation without the need to upload any files to the disk of the compromised host. You can obtain more information about Mimikatz/Metasploit integration from https://www.offensive-security.com/metasploit-unleashed/mimikatz/.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 3-2 lists a reference of these key topics and the page number on which each is found.

Table 3-2 Key Topics for Chapter 3

Key Topic Element	Description	Page Number
Paragraph	Defining and understanding what privilege escalation is	
List	Types of privilege escalation attacks	
List	Understanding the different types of cross-site scripting vulnerabilities	
Paragraph	Understanding SQL injection	
List	Identifying SQL injection categories	
Paragraph	Defining what DLL injection attacks are	
Paragraph	Understanding LDAP injection attacks	
Paragraph	Defining XML injection attacks	
Paragraph	Understanding pointer and object dereference	
Paragraph	Understanding directory (path) traversal attacks	
Paragraph	Defining and understanding what buffer overflows are	
Paragraph	Defining what an integer overflow is	
Paragraph	Understanding what a memory leak is	

Paragraph	Defining race conditions	
Paragraph	Understanding error handling vulnerabilities	
Paragraph	Identifying improper input handling vulnerabilities	
Paragraph	Examples of input validation vulnerabilities	
Paragraph	Understanding replay attacks	
Paragraph	Understanding session replay attacks	
Paragraph	Understanding cross-site request forgery attacks	
Paragraph	Understanding API-based attacks	
Paragraph	Defining resource exhaustion attacks	
Paragraph	Understanding memory leaks	
Paragraph	Understanding SSL stripping attacks	
Figure 3-4	An SSL Stripping Attack	
Paragraph	Understanding driver manipulation, including shimming and refactoring attacks	
Paragraph	Understanding the pass the hash attack	
Figure 3-5	The Pass the Hash Attack	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

privilege escalation

cross-site scripting

injection attacks

SQL injection (SQLi)

DLL injection

LDAP injection

XML injection

null pointer dereference

address space layout randomization (ASLR)

directory traversal

buffer overflow

integer overflows

memory leak

remote code execution (RCE)

race condition

time of check (TOC) or time of use (TOU)

error handling

input validation

black-box testing

white-box testing

compile-time errors

runtime error

structured exception handling (SEH)

input validation

replay attack

nonce

session replay

session hijacking

on-path attack

watering hole attack

cross-site request forgery (XSRF)

server-side request forgery (SSRF)

application programming interfaces (APIs)

resource exhaustion

memory leak

SSL stripping attack

shimming

refactoring

pass the hash attack

Mimikatz

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What tool is used by many penetration testers, attackers, and even malware that can be useful for retrieving password hashes from memory?
2. What is the act of restructuring driver code called?
3. What type of attack occurs when the attacker performs an MITM attack and can redirect a client to an insecure HTTP connect?
4. What is a modern framework of API documentation and development that is the basis of the OpenAPI Specification (OAS)?
5. What type of attack occurs when a user browsing the web is tricked into clicking something different than what the user thought he or she was clicking?
6. What type of attack is difficult to exploit because it takes advantage of the small window of time between when a service is used and its corresponding security control is executed in an application, operating system, or when temporary files are created?
7. What feature is supported in most modern operating systems that can help prevent the exploitation of buffer overflows, remote code execution, and memory corruption vulnerabilities?
8. What is a type of input validation vulnerability and attack against an application that parses XML input?

Chapter 4. Analyzing Potential Indicators Associated with Network Attacks

This chapter covers the following topics related to Objective 1.4 (Given a scenario, analyze potential indicators associated with network attacks) of the CompTIA Security+ SY0-601 certification exam:

- Wireless
  - Evil twin
  - Rogue access point
  - Bluesnarfing
  - Bluejacking
  - Disassociation
  - Jamming
  - Radio frequency identification (RFID)
  - Near-field communication (NFC)
  - Initialization vector (IV)
- On-path attack (previously known as man-in-the-middle attack/man-in-the-browser attack)
- Layer 2 attacks
  - Address resolution protocol (ARP) poisoning
  - Media access control (MAC) flooding
  - MAC cloning
- Domain name system (DNS)
  - Domain hijacking

    - DNS poisoning
    - Uniform resource locator (URL) redirection
    - Domain reputation
- Distributed denial of service (DDoS)
    - Network
    - Application
    - Operational technology (OT)
- Malicious code or script execution
    - PowerShell
    - Python
    - Bash
    - Macros
    - Visual Basic for Applications (VBA)

This chapter covers the potential indicators associated with wired and wireless network attacks. We start with an overview of attacks against wireless networks. Chapter 3, "Analyzing Potential Indicators Associated with Application Attacks," briefly covered on-path attacks (previously known as man-in-the-middle [MITM] or man-in-the-browser [MITB] attacks); however, this chapter covers these topics in more detail. You also learn about the most common attacks against Layer 2 devices and implementation, as well as attacks against DNS. In Chapter 3, you learned about denial-of-service (DoS) and resource exhaustion attacks. In this chapter, you learn what distributed denial-of-service (DDoS) attacks are. The chapter ends with an explanation of how attackers can leverage PowerShell, Python, Bash, macros, and Visual Basic for Applications (VBA) to create malicious code and perform script execution attacks.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review

Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 4-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 4-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Wireless Attacks	1
On-Path Attacks	2–3
Layer 2 Attacks	4
Domain Name System (DNS) Attacks	5
Distributed Denial-of-Service (DDoS) Attacks	6
Malicious Code or Script Execution Attacks	7

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which of the following is a type of related-key attack, which occurs when an attacker observes the operation of a cipher using several different keys

and finds a mathematical relationship between those keys, allowing the attacker to ultimately decipher data?

a. IV attack

b. Evil twin

c. WPA attack

d. None of these answers are correct.

2. Which of the following can be used to perform an on-path attack?

a. SQL injection

b. ARP cache poisoning

c. Buffer overflow

d. All of these answers are correct.

3. Which of the following describes when an attacker must first infect the victim's computer with a Trojan or a malicious browser extension or plug-in to intercept transactions from a web browser?

a. On-path attack

b. LDAP injection

c. SQL injection

d. Keylogger injection

4. Which of the following attacks occurs when the threat actor sends numerous unknown MAC addresses to a network switch to cause a DoS condition or to sniff network connections over a switched network while disrupting network performance?

a. CAM memory leak attack

b. MITM

c. ARP cache poisoning

d. MAC flooding attacks

5. Which of the following attacks occurs when the attacker changes the registration of a domain name without the permission of the original

owner/registrant?

a. Directory traversal

b. Path traversal

c. Dot-dot-slash

d. Domain hijacking

6. Which of the following protocols has been used for amplification attacks?

a. IPX

b. DNS

c. NetFlow

d. None of these answers are correct.

7. Which of the following has been used by attackers to create malicious macros in applications such as Excel or Word?

a. Keyloggers

b. VBA

c. DNS

d. NTP

Foundation Topics

Wireless Attacks

Attackers have performed attacks against wireless networks for many years. These attacks are relevant more than ever because of the advent of Internet of Things (IoT) and mobile devices. In the following sections, you learn the details about these attacks against wireless networks:

- Evil twin
- Rogue access point
- Bluesnarfing

- Bluejacking
- Disassociation
- Jamming
- Radio frequency identifier (RFID)
- Near-field communication (NFC)
- Initialization vector (IV)

Evil Twin Attacks

An ***evil twin*** is a rogue and unauthorized wireless access point (WAP) that uses the same service set identifier (SSID) name as a nearby wireless network, often public hotspots. Like the evil twin antagonist found in many sci-fi books, the device is identical in almost all respects to the authorized WAP. In Figure 4-1, the legitimate (corporate) access point is configured with the SSID corp-net, and the evil twin is deployed by an attacker configured with the same SSID.

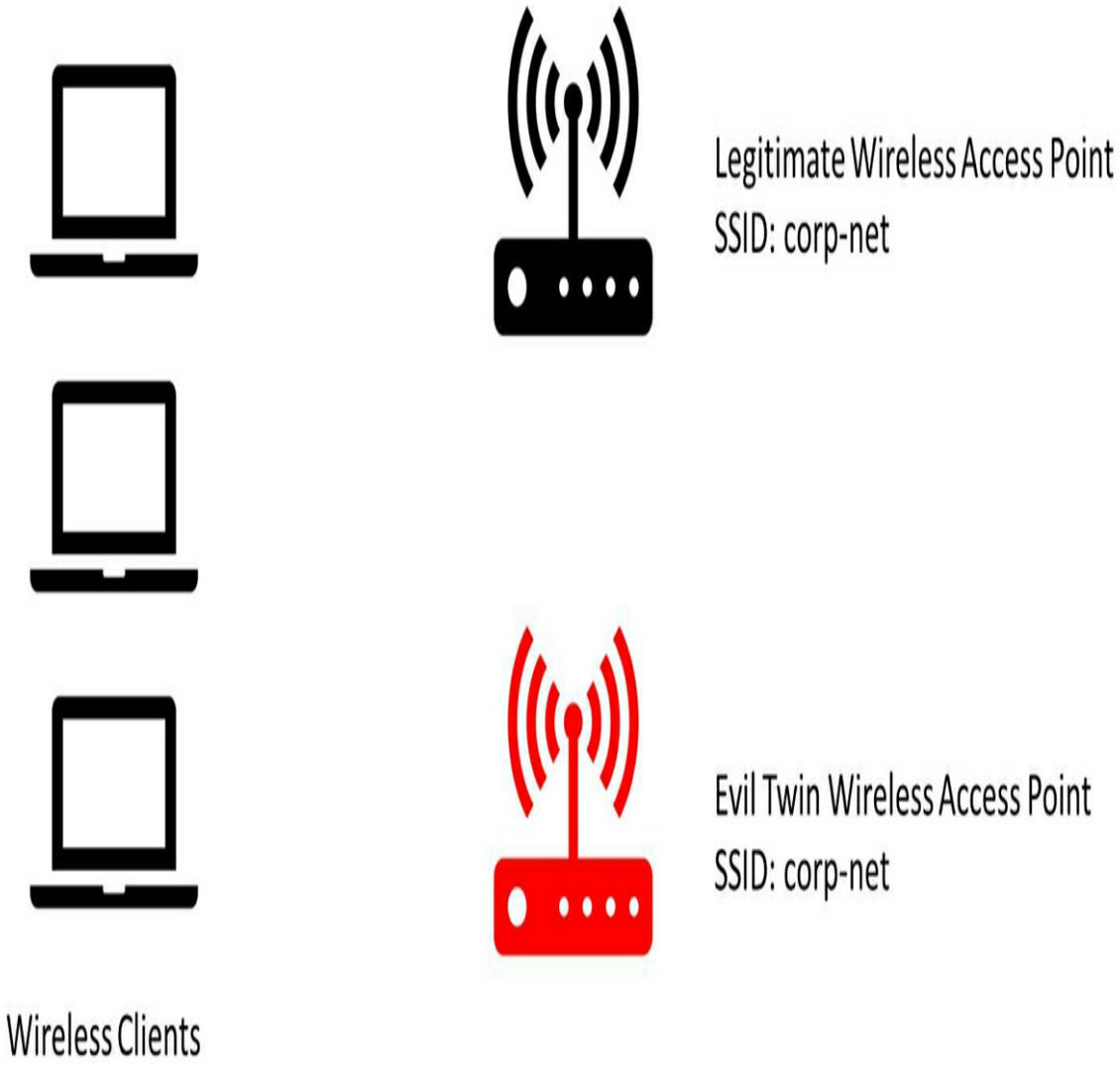


Figure 4-1 The Evil Twin Attack

While the antagonist in the sci-fi book usually has a beard or goatee, the WAP is controlled by a person with the same types of motives as the bearded evil twin. One of these motives is to steal sensitive information from wireless users. For example, an attacker might attempt to fool wireless users at an Internet café to connect to the counterfeit WAP to gain valuable information and passwords from the users. If the user is unlucky enough to connect to the evil twin, the attacker can easily record and digest all the information within the session later. This attack can also be enacted on organizations. To protect against this type of attack, organizations can implement virtual private

networks (VPNs) that require external authentication outside the WAP. Administrators should scan the network often for rogue APs that might be evil twins. Users in general should be trained not to send passwords, credit card numbers, and other sensitive information over wireless networks.

Rogue Access Points

Rogue APs can be described as unauthorized wireless access points/routers that allow access to secure networks. Sometimes companies lose track of the WAPs on their network. You should keep track of all your devices with network documentation. You also should use network mapping programs and Microsoft Visio to detect and document any rogue APs. Older WAPs, especially ones with weak encryption, should be updated, disabled, or simply disconnected from the network. Some companies may have a dozen WAPs and additional wireless devices such as repeaters, and it may be difficult to keep track of them. In this case, a network mapping program can be one of your best friends. In addition, you can search for rogue APs by using a laptop or handheld computer with the Windows wireless application, the wireless network adapter's built-in software, or third-party applications such as AirMagnet or NetStumbler. If traffic from a rogue AP does enter your network, a network-based intrusion detection system (NIDS) or network-based intrusion prevention system (NIPS) solution (covered in more detail in Chapter 19, "Implementing Secure Network Designs") can be instrumental in detecting and preventing that data and data that comes from other rogue devices. Organizations commonly perform site surveys to detect rogue APs and other unwanted wireless devices.

Bluesnarfing Attacks

Like any wireless technology, Bluetooth is vulnerable to attack. Bluejacking and bluesnarfing are two types of vulnerabilities to Bluetooth-enabled

devices. Bluetooth is also vulnerable to conflicts with other wireless technologies. For example, some WLAN (or Wi-Fi) standards use the 2.4-GHz frequency range, as does Bluetooth, and even though Bluetooth uses frequency hopping, conflicts can occur between 802.11g or 802.11b networks and Bluetooth personal area networks (PANs). To avoid this, you should use Bluetooth version 1.2 devices or greater, which employ adaptive frequency hopping, improving resistance to radio interference. Also, consider placing Bluetooth access points (if they are used) and WLAN access points in different areas of the building. Some companies have policies governing Bluetooth usage; in some cases, it is not allowed if 802.11 standards are in place, and in some cases, a company will enforce rules that say Bluetooth can be used only outside the building. In other cases, a company will put its 802.11 devices on specific channels or use WLAN standards that use the 5-GHz range.

Bluetooth-equipped devices can use near-field communication (NFC), which allows two mobile devices (or a mobile device and a stationary computer) to be automatically paired and transmit data. NFC is not limited to Bluetooth, but Bluetooth is probably the most common technology used to transmit data wirelessly over short distances. Of course, even though the distance is short, it can still be eavesdropped on. In addition, NFC is a data transmission protocol, but not necessarily secure. Data can be destroyed by use of a jammer, and users are also at risk of replay attacks. At the writing of this book, NFC does not offer preventive security in this regard, but a user can prevent these attacks by using only applications that offer SSL/TLS or other secure channels during an NFC session.

I know what you're thinking: the names of these attacks are starting to get a bit ridiculous! I guarantee the naming will improve as we progress through the rest of this book! Anyway, ***bluesnarfing*** is the unauthorized access of information from a wireless device through a Bluetooth connection. Generally, bluesnarfing is the theft of data (calendar information, phonebook contacts, and so on). It is possible to steal other information as well, but to pilfer any of this data, a pairing must be made between the attacking Bluetooth device and the Bluetooth victim. Ways of discouraging bluesnarfing include using a pairing key that is not easy to guess; for example, you should stay away from 0000 or similar default Bluetooth pairing keys! Otherwise, Bluetooth devices should be set to "undiscoverable"

(only after legitimate Bluetooth devices have been set up, of course), or Bluetooth can be turned off altogether, especially in areas that might be bluesnarfing playgrounds, such as Times Square in New York City. Some consider bluesnarfing to be a component of bluejacking, but for the Security+ exam, try to differentiate between the two.

Bluejacking Attacks

Bluejacking is the sending of unsolicited messages to Bluetooth-enabled devices such as mobile phones and tablets. Bluejacking is usually harmless, but if it does occur, it may appear that the Bluetooth device is malfunctioning. Originally, bluejackers would send only text messages, but with newer Bluetooth-enabled mobile devices, it is possible to send images and sounds as well. Bluejacking is used in less-than-reputable marketing campaigns. This type of attack can be stopped by setting the affected Bluetooth device to "undiscoverable" or by turning off Bluetooth altogether.

Disassociation and Deauthentication Attacks

DoS attacks can be run against WAPs in a variety of ways. One way is through the use of spoofed MAC addresses. If an attacker emulates a large number of wireless clients, each with a different spoofed MAC address, and never allows authentication for these clients to finish, then legitimate clients will no longer be able to be serviced by the WAP because all of the session spots will have been taken. This is a type of denial-of-service attack due to incomplete authentication. It can be prevented by configuring expiration timeouts for all sessions that have not had activity for a certain period of time. It can also be prevented by updating the WAP and implementing wireless frame protection. (Different providers have different names for this—for example, Cisco Management Frame Protection.)

Another type of wireless-based DoS is the ***Wi-Fi disassociation attack***—also

known as Wi-Fi deauthentication attack. The attacker targets a user who is connected to a Wi-Fi network by using a wireless scanner. Then the attacker forces that user's system to deauthenticate using special software and then reauthenticate to the WAP. By capturing the packets during the authentication process, the attacker can find out the SSID or ESSID of the WAP and possibly WPA/WPA2 handshake information. After that, the attacker will attempt to find out the WPA passphrase using a dictionary attack. This process sounds more difficult in theory than it actually is. Using automated software tools, an attacker can accomplish all this in under a minute. These types of "deauth" attacks can be prevented by using WPA2, by using complex passphrases, and by implementing technologies such as management frame protection by Cisco, which adds a hash value to management frames of information, such as the ones used by WPA/WPA2.

WP3 addresses the aforementioned issues using Simultaneous Authentication of Equals (SAE); however, no protocol is perfect. WPA3 also introduces other security vulnerabilities and concerns. For example, the WPA3 Dragonfly handshake used in different Wi-Fi networks is susceptible to different attacks. An attacker can use similar techniques against the EAP implementation and then use the WPA3 Dragonfly handshake to recover a user's password. You can obtain details about these attacks at https://wpa3.mathyvanhoef.com.

A WAP can fall victim to brute-force attacks if WPS is used. The key of the WAP can also be compromised by a brute-force attack known as an exhaustive key search. This type of attack can be prevented by limiting the number of times a password/passphrase can be tried, using time delays between attempts, and requiring complex answers during the authentication process. Also, as a corrective security control, attacking IP addresses can be blocked.

Jamming Attacks

The purpose of ***jamming*** wireless signals or causing wireless network interference is to create a full or partial DoS condition in the wireless

network. Such a condition, if successful, is very disruptive. Most modern wireless implementations provide built-in features that can help immediately detect such attacks. To jam a Wi-Fi signal or any other type of radio communication, an attacker basically generates random noise on the frequencies that wireless networks use. With the appropriate tools and wireless adapters that support packet injection, the attacker can cause legitimate clients to disconnect from wireless infrastructure devices.

Radio Frequency Identifier (RFID) Attacks

Radio frequency identification (RFID) has many uses, but it all boils down to identifying and tracking tags that are attached to objects. In the security world, that generally means authentication.

As with any wireless technology, RFID is susceptible to attack. For example, some RFID tags can be affected by skimming, on-path attacks, sniffing, eavesdropping/replaying, spoofing, and jamming (DoS). From an authentication standpoint, the attacker uses these attacks to try to find out the passcode. An RFID tag can also be reverse-engineered if the attacker gets possession of it. Finally, power levels can be analyzed to find out passwords. On some RFID tags, correct passcodes emit a different level of power than incorrect passcodes. To prevent these attacks, a security administrator (or team) should consider newer-generation RFID devices, encryption, chip coatings, filtering of data, and multifactor authentication methods. Encryption is one of the best methods. Included in this prevention method are rolling codes, which are generated with a pseudorandom number generator (PRNG), and challenge-response authentication (CRA), where the user (or user's device) must present the valid response to the challenge.

Note

As covered in Chapter 2, RFID ranges vary depending on the electromagnetic (EM) frequency band used—from 10 cm up to 200 meters. Many authentication readers can be as much as 1

meter, which is enough to facilitate skimming of information by attackers. One way to avoid this is to use newer RFID proximity readers and keys (ones that use lower frequencies) from respectable companies.

Near-Field Communication (NFC) Attacks

An alternative to RFID is to utilize the set of protocols called ***near-field communication (NFC)***. NFC generally requires that communicating devices be within 4 cm of each other, which makes skimming of information difficult. If an employee uses a smartphone to enter a building instead of an RFID device, then NFC should be implemented. NFC has obvious benefits for contactless payment systems or any other non-contact-oriented communications between devices. However, for optimal security, you should use contact-oriented readers and cards.

Initialization Vector (IV) Attacks

Initialization vector (IV) attacks are another vulnerability of wireless networks. An IV attack is a type of related-key attack, which occurs when an attacker observes the operation of a cipher using several different keys and finds a mathematical relationship between those keys, allowing the attacker to ultimately decipher data. An initialization vector is a random fixed-sized input that occurs in the beginning of every WEP or WPA packet. For WEP, the IV size was small (24-bit) and led to many successful attacks on WEP. This is why WEP is considered to be deprecated and insecure. The best way to prevent IV attacks is to use stronger wireless protocols such as WPA2 with AES and WPA3.

WEP is susceptible to many different attacks, and it is therefore considered an obsolete wireless protocol. WEP must be avoided, and many wireless

network devices no longer support it. WEP keys exist in two sizes: 40-bit (5-byte) and 104-bit (13-byte) keys. In addition, WEP uses a 24-bit IV, which is prepended to the preshared key (PSK). When you configure a wireless infrastructure device with WEP, the IVs are sent in plaintext.

WEP has been defeated for decades. WEP uses RC4 in a manner that allows an attacker to crack the PSK with little effort. The problem is how WEP uses the IVs in each packet. When WEP uses RC4 to encrypt a packet, it prepends the IV to the secret key before including the key in RC4. Subsequently, an attacker has the first 3 bytes of an allegedly "secret" key used on every packet. To recover the PSK, the attacker just needs to collect enough data from the air. The attacker can accelerate this type of attack by just injecting ARP packets (because the length is predictable), which allows him or her to recover the PSK much faster. After recovering the WEP key, the attacker can use it to access the wireless network. The attacker can also use the Aircrack-ng set of tools to crack (recover) the WEP PSK.

On-Path Attacks

In an ***on-path attack***, an attacker places himself or herself in-line between two devices or individuals who are communicating in order to eavesdrop or manipulate the data being transferred. As discussed earlier in this chapter, on-path attacks were previously known as man-in-the-middle (MITM) attacks.

On-path attacks can happen at Layer 2 or Layer 3 of the Open Systems Interconnection (OSI) model for computer networks.

Tip

If you are not familiar with the OSI model, the following article and embedded video describe it in detail: https://www.networkworld.com/article/3239677/the-osi-model-explained-and-how-to-easily-remember-its-7-layers.html.

Figure 4-2 demonstrates an on-path attack.

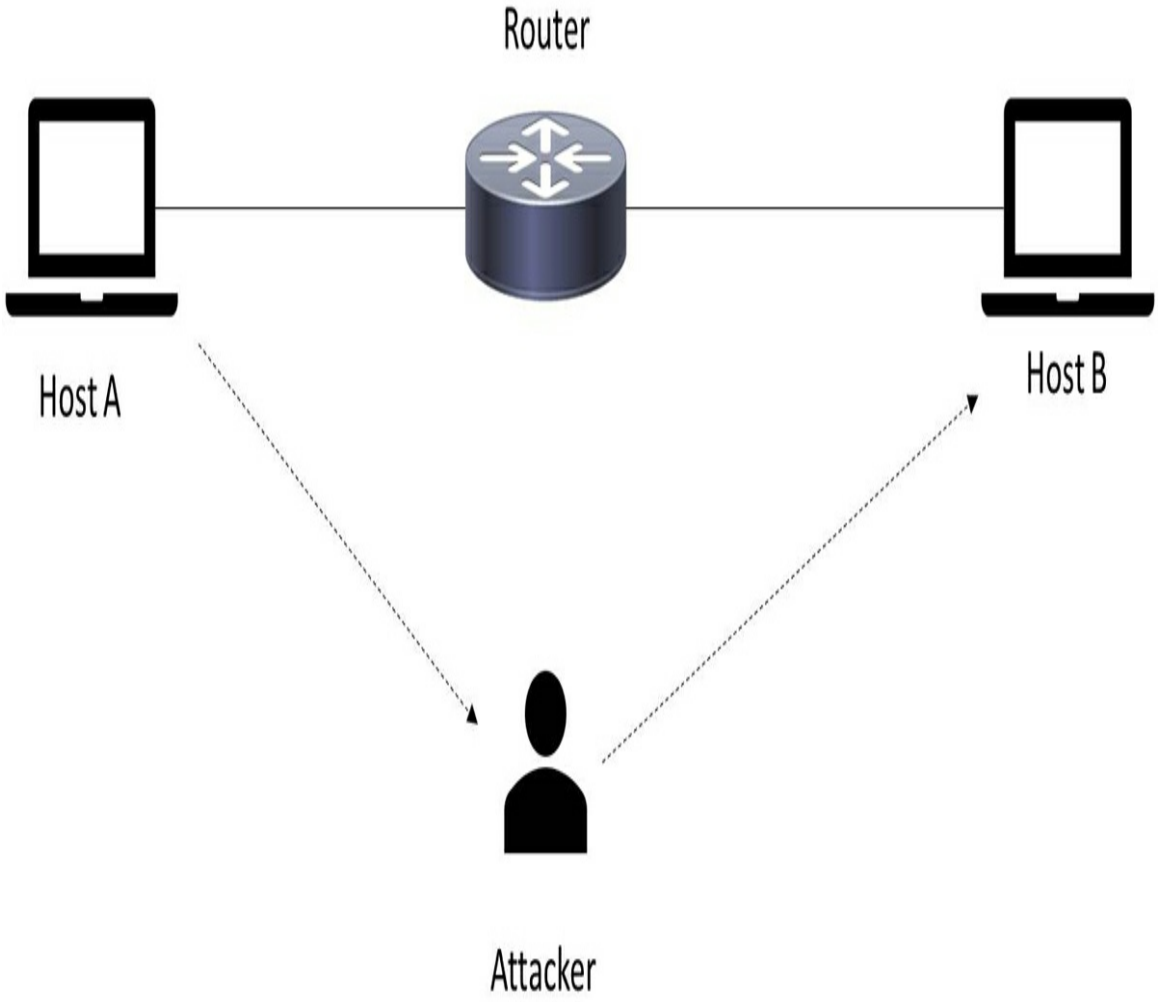


Figure 4-2 An On-Path Attack

In the figure the attacker intercepts the packets between host A and host B. On-path attacks could be used to sniff any unencrypted sensitive data or to interact with the user to potentially provide sensitive information to the attacker. ARP cache poisoning attacks can be used to perform on-path attacks. You learn about the ARP cache poisoning attacks later in this chapter.

On-path attacks are also often leveraged by attackers in wireless networks.

For instance, an attacker could impersonate a wireless access point or "provide free Internet access" in a coffee shop, hotel, airport, or any other public area. Then the attacker could steal information from the users connecting to such a fake wireless access point.

An attacker could also infect a victim's computer with a Trojan or a malicious browser extension or plug-in. The attacker usually gets the malware onto the victim's computer through some form of trickery or deceit. For example, the victim may have been asked to install some plug-in to watch a video or maybe update a program or install a screensaver. After the victim is tricked into installing malware onto his or her system, the malware simply waits for the victim to visit a targeted site. This browser-based on-path attack malware can invisibly modify transaction information like the amount or destination. It can also create additional transactions without the user knowing. Because the requests are initiated from the victim's computer, it is very difficult for the web service to detect that the requests are fake.

Layer 2 Attacks

Layer 2 switched environments (physical or virtual environments) can be targets for attackers. The following sections cover some of the most common Layer 2 attacks.

Address Resolution Protocol (ARP) Poisoning Attacks

ARP cache poisoning (also known as ARP spoofing) is an example of an attack that leads to an on-path attack scenario. An ARP spoofing attack can target hosts, switches, and routers connected to a Layer 2 network by poisoning the ARP caches of systems connected to the subnet and by intercepting traffic intended for other hosts on the subnet. In Figure 4-3, the attacker spoofs Layer 2 MAC addresses to make the victim (the host with the IP address 10.1.1.2) believe that the Layer 2 address of the attacker is the Layer 2 address of its default gateway (10.1.1.1).

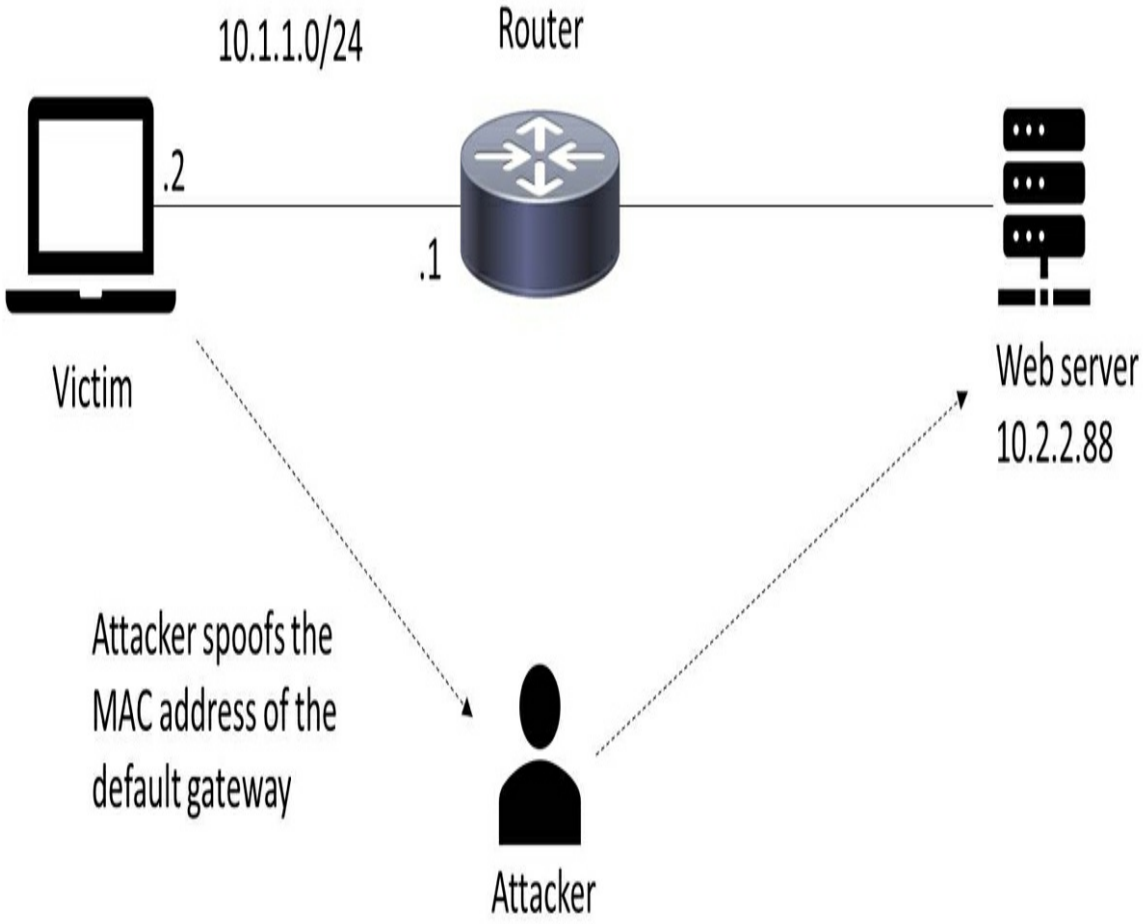


Figure 4-3 ARP Cache Poisoning Attack

The packets that are supposed to go to the default gateway are forwarded by the switch to the Layer 2 address of the attacker on the same network. The attacker can forward the IP packets to the correct destination to allow the client to access the web server (10.2.2.88).

A common mitigation for ARP cache poisoning attacks is to use Dynamic ARP Inspection (DAI) on switches to prevent spoofing of the Layer 2 addresses.

Another example of a Layer 2 on-path attack is to place a rogue switch in the network and use Spanning Tree Protocol (STP) to become the root switch. This could allow an attacker to see any traffic that needs to be sent through the root switch.

An attacker can carry out an on-path attack at Layer 3 by placing a rogue router on the network and then tricking the other routers into believing that this new router has a better path. It is also possible to perform an on-path attack by compromising the victim's system and installing malware that can intercept the packets sent by the victim. The malware can capture packets before they are encrypted if the victim is using SSL/TLS/HTTPS or any other mechanism.

Media Access Control (MAC) Flooding Attacks

Layer 2 switches flood frames destined to unknown destination MAC addresses and store the source address and port of every arriving packet in content addressable memory (CAM). All switches have an absolute amount of memory space for the number of MAC addresses that can be learned. Some can scale to millions of entries, whereas others do not. The forwarding table, however, has only a limited address space. Attackers can try to flood the forwarding table to cause a denial-of-service condition. In short, ***MAC flooding attacks*** send numerous unknown MAC addresses to a network switch to cause a DoS condition. In addition, once the Layer 2 forwarding table limit is exceeded, packets are flooded to all ports in a virtual LAN (VLAN). This, in turn, enables the attacker to sniff network connections over a switched network while disrupting network performance.

Tip

Port security is a dynamic feature that can be used to limit and identify the MAC addresses of hosts in a network and protect against this type of attack. When a network switch port is configured with port security and the maximum number of MAC addresses is reached, the MAC address of a host attempting to access the port that is different from any of the identified "secure MAC addresses" triggers a security alert to the administrator.

MAC Cloning Attacks

A ***MAC cloning attack*** (or MAC spoofing attack) occurs when an attacker sniffs the network for valid MAC addresses and then uses those MAC addresses to perform other actions. These actions could be to potentially bypass Layer 2 access control policies or masquerade as the default gateway to perform a on-path attack.

Best Practices to Protect Against Layer 2 Attacks

The following are some Layer 2 security best practices for securing your infrastructure:

- Select an unused VLAN (other than VLAN 1) and use it as the native VLAN for all your trunks. Do not use this native VLAN for any of your enabled access ports. Avoid using VLAN 1 anywhere because it is the default.
- Administratively configure switch ports as access ports so that users cannot negotiate a trunk; also disable the negotiation of trunking (that is, do not allow Dynamic Trunking Protocol [DTP]).
- Limit the number of MAC addresses learned on a given port with the port security feature.
- Control Spanning Tree to stop users or unknown devices from manipulating it. You can do so by using the BPDU Guard and Root Guard features.
- Turn off Cisco Discovery Protocol (CDP) on ports facing untrusted or unknown networks that do not require CDP for anything positive. (CDP operates at Layer 2 and might provide attackers information you would rather not disclose.)
- On a new switch, shut down all ports and assign them to a VLAN that is not used for anything other than a parking lot. Then bring up the ports and assign correct VLANs as the ports are allocated and needed.

- Use Root Guard to control which ports are not allowed to become root ports to remote switches.
- Use Dynamic ARP Inspection (DAI).
- Use IP Source Guard to prevent spoofing of Layer 3 information by hosts.
- Implement 802.1X when possible to authenticate and authorize users before allowing them to communicate to the rest of the network.
- Use DHCP snooping to prevent rogue DHCP servers from impacting the network.
- Use storm control to limit the amount of broadcast or multicast traffic flowing through a switch.
- Deploy access control lists, such as Layer 3 and Layer 2 ACLs, for traffic control and policy enforcement.

Domain Name System (DNS) Attacks

DNS is used practically everywhere. For example, every time that you visit a website, the domain name (for example, facebook.com, twitter.com, yourbank.com, and so on) is "resolved" to an IP address. In many cases, when machines communicate to each other using APIs, DNS is also used. Consequently, it is a big target for attackers. In addition to attacks against a DNS infrastructure, attackers have used DNS tunnels to steal data and registered domains to perform malicious tasks (such as exfiltrating data to a malicious domain, hosting malware, and allowing compromised devices to communicate with a command-and-control [C2] server). Let's start by examining how attackers can hijack a domain.

Domain Hijacking Attacks

Domain hijacking is a type of hijacking attack in which the attacker changes the registration of a domain name without the permission of the original

owner/registrant. One of the most common methods to perform domain hijacking is using social engineering. An attacker may pretend to be an employee of an organization or the domain registrar over a phone conversation to obtain access to the domain administration area. The attacker could also infect your system with malware (e.g., a keylogger or a Trojan) and steal the credentials for the domain admin.

To prevent domain hijacking, individuals and organizations should use trusted domain registrars. If the registrar provides you with the option to use multifactor authentication, use it! Many individuals and organizations select domain registrars that allow you to anonymize the administrative contact information about a domain (displayed in the WHOIS information).

Note

WHOIS is a free and publicly accessible directory from which domain names can be queried to discover contact and technical information behind registered domain names. You can access it at https://who.is/.

DNS Poisoning Attacks

DNS poisoning (or DNS cache poisoning) is the modification of name resolution information that should be in a DNS server's cache. It is done to redirect client computers to incorrect websites. This can happen through improper software design, misconfiguration of name servers, and maliciously designed scenarios exploiting the traditionally open architecture of the DNS system. Let's say a client wants to go to www.comptia.org. That client's DNS server will have a cache of information about domain names and their corresponding IP addresses. If CompTIA's site were visited in the recent past by any client accessing the DNS server, its domain name and IP should be in the DNS server's cache. If the cache is poisoned, it could be modified in such a way to redirect requests for www.comptia.org to a different IP address and website. This other site could be a phishing site or could be malicious in

some other way. This attack can be countered by using Transport Layer Security (TLS) and digital signatures or by using Domain Name System Security Extensions (DNSSEC), which uses encrypted electronic signatures when passing DNS information, and finally, by patching the DNS server. You might use a Transaction Signature (TSIG) to provide authentication for DNS database updates. This protocol uses shared secret keys and one-way hashing to provide security. One item of note: the hashing procedure might not be secure enough for your organization, so you may want to consider alternatives when updating DNS databases.

Unauthorized zone transfers are another bane to DNS servers. Zone transfers replicate the database that contains DNS data; they operate on top of TCP. If a zone transfer is initiated, say through a reconnaissance attack, server name and IP address information can be stolen, resulting in the attacker accessing various hosts by IP address. To defend against this, zone transfers should be restricted and audited in an attempt to eliminate unauthorized zone transfers and to identify anyone who tries to exploit the DNS server in this manner. Vigilant logging of the DNS server and the regular checking of DNS records can help detect unauthorized zone transfers.

A Windows computer’s hosts file can also be the victim of attack. The hosts file is used on a local computer to translate or resolve hostnames to IP addresses. This is the predecessor to DNS, and although the file is normally empty of entries, it is still read and parsed by Windows operating systems. Attackers may attempt to hijack the hosts file in an attempt to alter or poison it or to try to have the client bypass DNS altogether. The best defense for this is to modify the computer’s hosts file permissions to read-only. It is located at the following path: \%systemroot%\System32\drivers\etc.

If the file has already been hijacked, and you don’t use the file for any static entries, delete it, and Windows should re-create it automatically at the next system startup. If Windows does not, then you can easily re-create a standard hosts file by simply making a blank hosts.txt file and placing it in the path mentioned previously. Some people use the hosts file as a security measure as well. This is done by adding entries that redirect known bad domains to other safe locations or the localhost. Generally, this is done in conjunction with disabling the DNS client service. However, in general, the average Windows user requires the DNS client service.

Hosts files and vulnerable DNS software can also be victims of pharming attacks. As discussed in Chapter 1, ***pharming*** occurs when an attacker redirects one website's traffic to another website that is bogus and possibly malicious. Pharming can be prevented by carefully monitoring DNS configurations and hosts files. Unfortunately, if an ISP's DNS server is compromised, that will be passed on to all the small office/home office (SOHO) routers that the ISP services. So, it becomes more important for end users to be conscious of pharming. They can prevent it by turning on phishing and pharming filters within the browser and by being careful of which websites they access. Users can also check their local hosts files. By default, the file doesn't have any entries, so if they see entries and have never modified the file themselves, they should either delete the entries or delete the file entirely.

Although it is less of an actual attack, ***domain name kiting*** (or simply domain kiting) is the process of deleting a domain name during the five-day grace period (known as the add grace period, or AGP) and immediately reregistering it for another five-day period. This process is repeated any number of times with the end result of having the domain registered without ever actually paying for it. It is a malicious attack on the entire Domain Name System that misuses the domain-tasting grace period. The result is that a legitimate company or organization often cannot secure the domain name of its choice.

As you can see, the DNS server can be the victim of many attacks due to its visibility on the Internet. It should be closely monitored at all times. Other highly visible servers such as web servers and mail servers should be likewise monitored, audited, and patched as soon as updates are available.

Uniform Resource Locator (URL) Redirection Attacks

Attackers can use social engineering and other techniques to redirect users from a trusted website to a malicious site (***URL redirection attacks***). These techniques include phishing, spear phishing, pharming, and others.

In Chapter 3, you learned about cross-site scripting (XSS) and how attackers can leverage those vulnerabilities to redirect a user to a malicious site. XSS is a flaw in the web application or a website, but the victim is the actual user of such application. These types of attacks are typically the entry point for an attacker into an organization or a user's sensitive data. By validating the input of URLs passed and using whitelists, you can mitigate URL redirection.

Domain Reputation

Domain reputation is a technique used to validate the authenticity of a domain and the services using such domain (including websites and email messages). Domain reputation is used to track known malicious domains that point to websites hosting malware or those that are used for spam, phishing, spear phishing, and other malicious activities. Several technologies and standards have been created for domain authentication and validation. The first example we cover here is the ***Domain Keys Identified Mail (DKIM)***. DKIM is an industry standard defined in RFC 5585, and it provides a means for gateway-based cryptographic signing of outgoing messages. This allows you to embed verification data in an email header and for email recipients to verify the integrity of the email messages.

DKIM uses DNS TXT records to publish public keys. A few additional specifications related to DKIM exist. RFC 6376 introduces DKIM signatures; RFC 5863 provides information on DKIM development, deployment, and operation; and RFC 5617 addresses Author Domain Signing Practices (ADSP).

Another technology is the Sender Policy Framework (SPF), which enables recipients to verify the sender's IP addresses by looking up DNS records that list authorized mail gateways for a particular domain. SPF is an industry standard defined in RFC 4408. SPF uses DNS TXT resource records. Commercial products such as the Cisco Email Security Appliance (ESA) support SPF to verify HELO/EHLO and MAIL FROM identity (FQDN). When you enable SPF, the appliance adds headers in the message, allowing you to obtain additional intelligence on the email sender. One challenge is

that the effectiveness of SPF implementations is based on participation. Also, you need to invest time to make sure SPF records are up to date. Many organizations implement SPF because some of the most prevalent email providers nowadays do not accept mail without SPF records.

Some organizations go to the extent of not allowing any emails that do not have an SPF record. Doing so will block more spam emails; however, some legitimate mail might also be dropped if the sending entity hasn't configured SPF correctly.

Then there is also ***Domain-based Message Authentication***, ***Reporting & Conformance (DMARC)***. DMARC is a standard that was designed to thwart spammers from spoofing your domain to send email. Spammers can counterfeit the "From" address on an email message so that it appears to come from a user in your domain. For example, attackers can spoof your bank's domain name (that is, accounts@yourbank.com) to send you a fraudulent email in an effort to obtain your account information. DMARC is designed to block these emails before they appear in your email inbox. DMARC also provides visibility and reports into who is sending email on behalf of your domain to make sure that only legitimate emails are received.

DMARC is an open and free technology, allowing organizations and individuals to secure their domain's emails and maintain control of email delivery.

Distributed Denial-of-Service (DDoS) Attacks

A ***distributed denial-of-service (DDoS) attack*** occurs when a group of compromised systems attacks a single target, causing a DoS to occur at that host. A DDoS attack often utilizes a ***botnet***, which is a large group of computers known as robots or simply "bots." Often, these are systems owned by unsuspecting users. The computers in the botnet that act as attackers are known as zombies. An attacker starts the DDoS attack by exploiting a single vulnerability in a computer system and making that computer the zombie master, or DDoS master. The master system communicates with the other systems in the botnet. The attacker often loads malicious software on many

computers (zombies). The attacker then can launch a flood of attacks by all zombies in the botnet with a single command. DDoS attacks and botnets are often associated with exploit kits (such as the Blackhole kit) and ransomware.

DoS and DDoS attacks are difficult to defend against. Other than the methods mentioned previously in the DoS section, these attacks can be prevented to some extent by updated stateful firewalls, switches, and routers with access control lists; intrusion prevention systems (IPSs); and proactive testing. Several companies offer products that simulate DoS and DDoS attacks. By creating a test server and assessing its vulnerabilities with simulated DoS tests, you can find holes in the security of your server before you take it live. A quick web search for "DoS testing" shows a few of these simulation test companies.

An organization could also opt for a "clean pipe," which attempts to weed out DDoS attacks, among other attacks. This solution is offered as a service by Verisign and other companies. Manual protection of servers can be a difficult task; to implement proper DDoS mitigation, your organization might want to consider anti-DDoS technology and emergency response from an outside source or from the organization's cloud-based provider. Finally, if you do realize that a DDoS attack is being carried out on your network, call your ISP and request that this traffic be redirected.

One specific type of DDoS is the ***DNS amplification attack***. Amplification attacks generate a high volume of packets ultimately intended to flood a target website. In the case of a DNS amplification attack, the attacker initiates DNS requests with a spoofed source IP address. The attacker relies on *reflection*; responses are not sent back to the attacker but are instead sent "back" to the victim server. Because the DNS response is larger than the DNS request (usually), it *amplifies* the amount of data being passed to the victim. An attacker can use a small number of systems with little bandwidth to create a sizable attack. However, a DNS amplification attack can also be accomplished with the aid of a botnet, which has proven to be devastating to sections of the Internet during the period when the attack was carried out.

Note

Other protocols, such as the Network Time Protocol (NTP), have also been used to perform amplification attacks.

The primary way of preventing this attack is to block spoofed source packets. It can also be prevented by blocking specific DNS servers, blocking open recursive relay servers, rate limiting, and updating your own DNS server(s) often. Finally, as previously mentioned, you can make use of the DNSSEC, which are specifications that provide for origin authentication and data integrity.

Network DDoS attacks aim to target network infrastructure resources (for example, bandwidth, CPU, and memory utilization of the underlying network infrastructure). ***Application DDoS attacks*** target the resources of Layer 7 applications and often leverage known vulnerabilities against specific software. For example, an attacker can exploit denial-of-service vulnerabilities in web servers like Apache HTTP Server ("httpd") and NGINX or a specific web-based application.

Operational technology (OT) is the term used to describe physical items that can be programmed and connected to a network or the Internet. Typically, these devices are used to control electrical grids, pipelines, automobiles, manufacturing plant robots, and other critical infrastructure. Threat actors are always trying to find ways to launch denial-of-service attacks against OT environments and critical infrastructure.

At the same time, Internet of Things devices such as video doorbells, smart thermostats, IP-enabled cameras, and others have been compromised and used to launch very effective DDoS attacks against numerous organizations. A good example is the Mirai botnet that was used to launch a major DDoS several years ago against several high-profile companies and organizations.

Malicious Code or Script Execution Attacks

Attackers have used scripting languages such as ***PowerShell***, ***Python***, and ***Bash*** to perform enumeration, create exploits, and also employ postexploitation techniques. Numerous pieces of malware have used these legitimate tools, as well as ***macros*** embedded in Microsoft Word or Excel documents. Malicious macros are usually placed in documents and emailed to users in the hopes that the users will open the documents, thus executing it.

Visual Basic for Applications (VBA) is an event-driven programming capability in Microsoft operating systems and applications. When you write code using VBA, such code is compiled to a Microsoft proprietary pseudo-code. Applications such as Excel, Word, PowerPoint, and Outlook store this code as a separate stream in COM Structured Storage files—for example, in files such as .xls, .doc, .docx, and .pptx. Attackers have used VBA to create malicious macros and embed them in Excel, Word, and PowerPoint documents. These malicious macros have been designed to steal sensitive information, install keyloggers and Trojans, and perform other nefarious activities.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 4-2 lists a reference of these key topics and the page number on which each is found.

Table 4-2 Key Topics for Chapter 4

Key Topic Element	Description	Page Number
Paragraph	Defining and understanding evil twin attacks	
Paragraph	Defining and understanding what rogue wireless access points are	
Paragraph	Understanding bluesnarfing attacks	
Paragraph	Understanding bluejacking attacks	
Paragraph	Surveying disassociation and deauthentication attacks	
Paragraph	Defining wireless jamming attacks	
Paragraph	Understanding attacks against RFID implementations	
Paragraph	Understanding attacks against NFC implementations	

Section	Initialization Vector (IV) Attacks	
Figure 4-2	Understanding On-Path Attacks	
Paragraph	Understanding ARP poisoning attacks	
Paragraph	Understanding the implications of MAC flooding attacks	
Paragraph	Defining MAC cloning attacks	
Paragraph	Understanding domain hijacking attacks	
Paragraph	Defining DNS poisoning attacks	
Paragraph	Surveying URL redirection attacks	
Paragraph	Defining domain reputation	
Paragraph	Understanding DDoS attacks	
Paragraph	Understanding the difference between network and application-targeted DDoS attacks	
Paragraph	Surveying OT-related attacks	
Section	Malicious Code or Script Execution Attacks	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

evil twin

rogue APs

bluesnarfing

bluejacking

Wi-Fi disassociation attack

jamming

radio frequency identification (RFID)

near-field communication (NFC)

initialization vector (IV) attack

on-path attack

ARP cache poisoning

MAC flooding attacks

MAC cloning attack

domain hijacking

DNS poisoning

pharming

domain name kiting

URL redirection attacks

domain reputation

Domain Keys Identified Mail (DKIM)

Domain-based Message Authentication

Reporting & Conformance (DMARC)

distributed denial-of-service (DDoS) attack

botnet

DNS amplification attack

network DDoS attacks

application DDoS attacks

operational technology (OT)

PowerShell

Python

Bash

macros

Visual Basic for Applications (VBA)

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What is a Microsoft scripting language that attackers have used to perform postexploitation activities such as privilege escalation or to enumerate users?
2. What is a standard that was designed to thwart spammers from spoofing your domain to send email?
3. What is a web application vulnerability that could allow an attacker to perform a URL redirection attack?

4. What is the modification of name resolution information that should be in a DNS server's cache?

5. In what type of attack does the attacker sniff the network for valid MAC addresses and then use those MAC addresses to perform other actions?

Chapter 5. Understanding Different Threat Actors, Vectors, and Intelligence Sources

This chapter covers the following topics related to Objective 1.5 (Explain different threat actors, vectors, and intelligence sources) of the CompTIA Security+ SY0-601 certification exam:

- Actors and threats
  - Advanced persistent threat (APT)
  - Insider threats
  - State actors
  - Hacktivists
  - Script kiddies
  - Criminal syndicates
  - Hackers
    - Authorized
    - Unauthorized
    - Semi-authorized
  - Shadow IT
  - Competitors
- Attributes of actors
  - Internal/external
  - Level of sophistication/capability
  - Resources/funding

    - Intent/motivation
- Vectors
    - Direct access
    - Wireless
    - Email
    - Supply chain
    - Social media
    - Removable media
    - Cloud
- Threat intelligence sources
    - Open source intelligence (OSINT)
    - Closed/proprietary
    - Vulnerability databases
    - Public/private information sharing centers
    - Dark web
    - Indicators of compromise
    - Automated indicator sharing (AIS)
        - Structured threat information eXpression (STIX)/Trusted automated eXchange of indicator information (TAXII)
    - Predictive analysis
    - Threat maps
    - File/code repositories
- Research sources
    - Vendor websites
    - Vulnerability feeds
    - Conferences

- Academic journals
- Request for comments (RFC)
- Local industry groups
- Social media
- Threat feeds
- Adversary tactics, techniques, and procedures (TTP)

This chapter covers the different types and attributes of threat actors. You also learn what threat intelligence is, what it is used for, and the different types of threat intelligence sources and formats. The chapter concludes with a list of research resources that individuals and organizations can use to understand today's threats and to protect their networks and systems.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 5-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 5-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Actors and Threats	1–2
Attributes of Actors	3–4
Attack Vectors	5–6
Threat Intelligence and Threat Intelligence Sources	7–8
Research Sources	9–10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which of the following threat actors typically aims to cause denial-of-service conditions or deface websites due to a political or social belief?

a. Hacktivists

b. Semi-authorized hackers

c. Script kiddies

d. Nation state actors

2. Which of the following threat actors is typically motivated by money? (Choose the best answer.)

a. Authorized hackers

b. Criminal syndicates and organized crime

c. Exploit groups

d. None of these answers are correct.

3. Which of the following attributes make threat actors different?

a. Level of sophistication

b. Resources

c. Funding

d. All of these answers are correct.

4. Which incident response concept is designed to represent a cybersecurity incident and is made up of four parts?

a. FIRST

b. InfraGard

c. Diamond Model of Intrusion

d. All of these answers are correct.

5. Which of the following is an effective attack vector where an attacker could modify or tamper hardware or software from a vendor to perform mass compromise attacks?

a. Supply chain

b. Removable media

c. Wireless

d. Direct access

6. Which of the following elements could an attacker leverage to perform a cloud-based attack?

a. Misconfigured VMs

b. Unpatched applications and operating systems

c. Misconfigured storage buckets

d. All of these answers are correct.

7. Which term refers to the knowledge about an existing or emerging threat to assets, including networks and systems?

a. Threat intelligence

b. Threat feed

c. Threat model

d. None of these answers are correct.

8. Which technique is used when you leverage public information from DNS records, social media sites, websites, search engines, and other sources for reconnaissance?

a. OSINT

b. Threat maps

c. Threat models

d. Threat analysis

9. Which of the following could be threat and vulnerability research sources?

a. Vendor websites

b. Threat feeds

c. Vulnerability feeds

d. All of these answers are correct.

10. Which framework was created by MITRE to describe adversary tactics and techniques?

a. InfraGard

b. CVE

c. ATT&CK

d. CWE

Foundation Topics

Actors and Threats

It’s important to understand the types of attackers that you might encounter and their characteristics and personality traits. Keep in mind that these “actors” can find ways to access systems and networks just by searching the Internet. Some attacks can be perpetrated by people with little knowledge of computers and technology. However, other actors have increased knowledge, and with knowledge comes power—the type of power you want to be ready for in advance. However, all levels of attackers can be dangerous. Let’s look at several of the actors now.

The first of these personas is the ***script kiddie***. This unsophisticated individual has little or no skills when it comes to technology. This person uses code that was written by others and is freely accessible on the Internet. For example, a script kiddie might copy a malicious PHP script directly from one website to another; only the knowledge of “copy and paste” is required. This derogatory term is often associated with juveniles. Though the processes they use are simple, script kiddies can knowingly (and even unwittingly) inflict incredible amounts of damage to insecure systems. These people almost never have internal knowledge of a system and typically have a limited amount of resources, so the amount, sophistication, and extent of their attacks are constrained. They are often thrill-seekers and are motivated by a need to increase their reputation in the script kiddie world.

Next is the ***hacktivist***—a combination of the terms *hack* and *activist*. As with the term *hacker*, the name *hacktivist* is often applied to different kinds of activities—from hacking for social change, to hacking to promote political agendas, to full-blown cyberterrorism. Due to the ambiguity of the term, a hacktivist could be inside a company or attack from the outside and will have varying amounts of resources and funding. However, the hacktivist is usually far more competent than a script kiddie.

Cybercriminals might work on their own, or they might be part of ***criminal syndicates*** and ***organized crime***—a centralized enterprise run by people motivated mainly by money. Individuals who are part of an organized crime group are often well funded and can have a high level of sophistication.

Individuals might also carry out cyber attacks for governments and nation states. In this case, a government—and its processes—is known as an

advanced persistent threat (APT), though this term can also refer to the set of computer-attacking processes themselves. Often, an APT entity has the highest level of resources, including open-source intelligence (OSINT) *and* covert sources of intelligence. This, coupled with extreme motivation, makes the APT one of the most dangerous foes. In many cases, APTs are tied to nation ***state actors***. It is not a taboo anymore that governments also use cyber techniques as a method of warfare. State actors can use adversarial techniques to steal intellectual property, cause disruption, and potentially compromise critical infrastructure.

Note

OSINT is covered in more detail later in this chapter.

Companies should be prepared for insiders and competitors as well. In fact, they might be one and the same. An insider threat can be one of the deadliest to servers and networks, especially if the person making this threat is a system administrator or has security clearance at the organization. Not all "insiders" may be malicious, however. Think about the risks that ***shadow IT*** may introduce in a company. In shadow IT an employee or a group of employees use IT systems, network devices, software, applications, and services without the approval of the corporate IT department. For example, an engineer may just deploy a physical or virtual server and host his or her own application in it. The engineer may not patch or monitor that system often for any security vulnerabilities. Subsequently, threat actors may easily compromise the vulnerable system or application. This problem has grown exponentially in recent years because more companies and individuals are adopting cloud-based applications and services. It is very easy to "spin off" a new virtual machine (VM), container, or storage element in the cloud. Employees often put sensitive corporate information in cloud services that are often not closely monitored, introducing a significant risk to the corporation.

You should know these actors and their motivations, but in the end, it doesn't matter too much what you call the attackers. Be ready to prevent the basic attacks and the advanced attacks and treat them all with respect.

Not all hackers are malicious, however. That's right! There are different

types of hackers. Different organizations use various names, but some of the common labels include the following:

- An ***authorized hacker*** (white hat) is also known an ethical hacker; this term describes those who use their powers for good. They are often granted permission to hack.
- An ***unauthorized hacker*** (black hat) is a malicious hacker who may have a wide range of skill, but this category is used to describe those hackers involved with criminal activities or malicious intent.
- A ***semi-authorized*** (gray hat) ***hacker*** falls somewhere in the middle of authorized and unauthorized. This person doesn't typically have malicious intent but often runs afoul of ethical standards and principles.

Attributes of Threat Actors

Now that you have learned about the different types of threat actors, let's try to understand their attributes, ***intent***, and ***motivation***. The intent and motivation, in most cases, differ depending on the type of attacker. For example, the motivation for hacktivists is typically around protesting against something that they do not agree with (a political or religious belief, human rights, etc.). Hacktivists tend to use disruption techniques, perform denial-of-service (DoS) attacks, or deface websites. The intent of criminals is typically to make money. Nation state actors can also have different motivations and intent. Some nation state actors just want to steal intellectual property from another nation, government, or a corporation; whereas others want to cause disruption and bring down critical infrastructure. Another example is an ***internal actor*** (insider) who in some cases could be motivated by money being offered from an ***external actor*** (either to steal information or to cause disruption). The internal actor could also be a disgruntled employee who may leave a backdoor either to later access confidential information or to trigger a denial-of-service condition on critical systems and applications.

Another difference between the different types of threat actors is the level of sophistication/capability and the resources/funding. Obviously, a disgruntled

employee, a hacktivist, or an amateur script kiddie will not have the resources and funding that nation state actors may have at their disposal. Criminal organizations in some cases may have more resources, money, and access to sophisticated exploits than hacktivists.

Active intrusions start with an adversary who is targeting a victim. The adversary will use various capabilities along some form of infrastructure to launch an attack against the victim. Capabilities can be various forms of tools, techniques, and procedures, while the infrastructure is what connects the adversary and victim.

Attack Vectors

A threat actor may leverage different attack vectors:

- **Direct access:** Direct network access or physical access to a device provides one type of attack vector.
- **Wireless:** This attack vector includes hijacking wireless connections, rogue wireless devices, evil twins, and the attacks you learned in Chapter 4, "Analyzing Potential Indicators Associated with Network Attacks."
- **Email:** Email attack vectors include phishing and spear phishing emails with malicious attachments or malicious links.
- **Supply chain:** One of the most effective attacks for mass compromise is to attack the supply chain of a vendor to tamper with hardware and/or software. This tampering might occur in-house or earlier, while in transit through the manufacturing supply chain.
- **Social media:** This attack vector includes leveraging social media platforms for reconnaissance or to launch social engineering attacks.
- **Removable media:** Many attackers have successfully compromised victim systems by just leaving USB sticks (sometimes referred to as USB keys or USB pen drives) unattended or placing them in strategic locations. Often users think that the devices are lost and insert them into

their systems to figure out who to return the devices to; before they know it, they may be downloading and installing malware. Plugging in that USB stick you found lying on the street outside your office could lead to a security breach.

- **Cloud:** Attackers can leverage misconfigured and insecure cloud deployments including unpatched applications, operating systems, and storage buckets.

Threat Intelligence and Threat Intelligence Sources

Threat intelligence refers to knowledge about an existing or emerging threat to assets, including networks and systems. Threat intelligence includes context, mechanisms, ***indicators of compromise*** (IoCs), implications, and actionable advice. This type of intelligence includes specifics on the tactics, techniques, and procedures of these adversaries. The primary purpose of threat intelligence is to inform business decisions regarding the risks and implications associated with threats.

Converting these definitions into common language could translate to threat intelligence being evidence-based knowledge of the capabilities of internal and external threat actors. This type of data can be beneficial for the security operations center (SOC) of any organization. Threat intelligence extends cybersecurity awareness beyond the internal network by consuming intelligence from other sources Internet-wide related to possible threats to you or your organization. For instance, you can learn about threats that have impacted different external organizations. Subsequently, you can proactively prepare rather than react after the threat is seen against your network. One service that threat intelligence platforms typically provide is an enrichment data feed.

During the investigation and resolution of a security incident, you may also need to communicate with outside parties regarding the incident. Examples include but are not limited to contacting law enforcement, fielding media inquiries, seeking external expertise, and working with Internet service providers (ISPs), the vendor of your hardware and software products, threat

intelligence vendor feeds, coordination centers, and members of other incident response teams.

You can also share relevant incident IoC information and other threat intelligence with industry peers. There are ***public*** and ***private information sharing centers***. A good example of information-sharing communities includes the Financial Services Information Sharing and Analysis Center (FS-ISAC).

Tip

Information Sharing and Analysis Centers (ISACs) are private-sector critical infrastructure organizations and government institutions that collaborate and share information between each other. ISACs exist for different industry sectors. Examples include automotive, aviation, communications, IT, natural gas, elections, electricity, financial services, health-care, and many other ISACs. You can learn more about ISACs at www.nationalisacs.org.

A cybersecurity incident response plan should account for these types of interactions with outside entities. It should also include information about how to interact with your organization's public relations department, legal department, and upper management. You should also get their buy-in when sharing information with outside parties to minimize the risk of information leakage. In other words, you should avoid leaking sensitive information regarding security incidents with unauthorized parties. These actions could potentially lead to additional disruption and financial loss. You should also maintain a list of all the contacts at those external entities, including a detailed list of all external communications for liability and evidentiary purposes.

You can also subscribe to commercial and open-source threat intelligence feeds. Some of these feeds could be *closed* or *proprietary*. In cybersecurity, there is a concept called ***open-source intelligence (OSINT)***. OSINT applies to offensive security (ethical hacking/penetration testing) and defensive security. In offensive security, OSINT enables you to leverage public

information from DNS records, social media sites, websites, search engines, and other sources for reconnaissance—in other words, to obtain information about a targeted individual or an organization. When it comes to threat intelligence, OSINT refers to public and free sources of threat intelligence.

Tip

The following GitHub repository includes numerous references about OSINT tools and sources: https://github.com/The-Art-of-Hacking/h4cker/tree/master/osint.

You also could leverage other sources to obtain information about threats, including the ***dark web***. The "deep web" is the part of the Internet that has not been indexed by search engines like Google or Bing. The dark web is just a subset of what is considered the "deep web" and is the place where many threat actors perform malicious activities such as selling stolen credit card numbers, health records, and other personal information. Criminals also sell drugs, guns, counterfeit equipment, and exploits. In some cases, security professionals go to the dark web to perform research and try to find different threats and exploits that could affect their organization. This task is easier said than done. This is why several companies sell "dark web monitoring" and threat intelligence services.

As a security professional, you also have to pay attention to ***vulnerability databases*** and feeds. One of the most popular vulnerability databases is the United States National Vulnerability Database (NVD), part of the National Institute of Standards and Technology (NIST). You can access the NVD at https://nvd.nist.gov.

The NVD provides information about vulnerabilities disclosed worldwide and is assigned the Common Vulnerability and Exposure (CVE) identifiers. CVE is a standard created by MITRE (www.mitre.org) that provides a mechanism to assign an identifier to vulnerabilities so that you can correlate the reports of those vulnerabilities among sites, tools, and feeds. You can obtain additional information about CVE at https://cve.mitre.org.

When it comes to threat intelligence, several standards and formats allow you to perform ***automated indicator sharing (AIS)*** within your organization or

with industry peers. The next section covers the most popular standards and formats for threat intelligence exchange.

Structured Threat Information eXpression (STIX) and the Trusted Automated eXchange of Indicator Information (TAXII)

The ***Structured Threat Information eXpression (STIX)*** is an OASIS standard designed for sharing of cyber-attack information. STIX details can contain data such as the IP addresses or domain names of command-and-control servers (often referred to C2 or CnC), malware hashes, and so on. STIX was originally developed by MITRE and is now maintained by OASIS.

The ***Trusted Automated eXchange of Indicator Information (TAXII)*** is also an OASIS standard that provides an open transport mechanism that standardizes the automated exchange of cyber-threat information. TAXII was originally developed by MITRE and is now maintained by OASIS.

Note

You can access the STIX and TAXII specification details at https://oasis-open.github.io/cti-documentation.

In short, STIX is the actual threat intelligence document in JSON format and TAXII is the transport mechanism.

Example 5-1 provides a STIX threat intelligence document.

Example 5-1 A STIX Threat Intelligence Document

```
{
    "type": "bundle",
    "id": "bundle--aaaa-bbbb-cccc-1234-123123123",
    "objects": [
```

```
        {
            "type": "indicator",
            "spec_version": "2.1",
            "id": " indicator-aaabbbcc-b0123-cbc ",
            "created": "2020-10-21T11:09:08.123Z",
            "modified": "2020-10-21T11:09:08.123Z",
            "name": "Windows Trojan
Trojan/Win32.PornoAsset.R39927",
            "description": "This file hash indicates that a sample
of
 Trojan/Win32.PornoAsset.R39927 is present.",
            "indicator_types": [
                "malicious-activity"
            ],
            "pattern": "[file:hashes.'SHA-256' =
 '0f1f19244fcc11818083aa1f943bbead338f89e046b8a57a50ec7cf48b62496f
']",
            "pattern_type": "stix",
            "valid_from": "2020-03-11T08:00:00Z"
        },
        {
            "type": "malware",
            "spec_version": "2.1",
            "id": " malware--aaaa-bbb-cccc-dddd-eeee ",
            "created": "2020-10-21T11:09:08.123Z",
            "modified": "2020-10-21T11:09:08.123Z",
            "name": "Poison Ivy",
            "malware_types": [
                "remote-access-trojan"
            ],
            "is_family": false
        },
        {
            "type": "relationship",
            "spec_version": "2.1",
            "id": "relationship--12344-4334-1253-6644-b2455",
            "created": "2020-10-21T11:09:08.123Z",
            "modified": "2020-10-21T11:09:08.123Z",
            "relationship_type": "indicates",
```

```
            "source_ref": "indicator-aaabbbcc-b0123-cbc",
            "target_ref": "malware--aaaa-bbb-cccc-dddd-eeee"
        }
    ]
}
```

Predictive analysis and predictive analytics can help discover security threats in your network. Machine-learning-powered solutions have evolved during the last few years, providing benefits that extend beyond signature-based intrusion detection and prevention. Many organizations have to handle large volumes of data from multiple security devices and sources. This makes it almost impossible for individuals to keep up and analyze. As described in Chapter 2, "Analyzing Potential Indicators to Determine the Type of Attack," machine learning (ML) provides a way to quickly analyze security logs and data to perform advanced calculations on it in near real time and help respond to security incidents.

Several organizations have created ***threat maps*** to try to visualize the types of attacks and threats in their network and across the Internet. A cyber-threat map (also known as an attack map) is a real-time map of security attacks happening at any given time. These maps are visualizations from data aggregated from many different sensors around the world. Threat maps help illustrate how prevalent cyber attacks are.

Another source of threat and security information is ***file and code repositories***. Threat actors often release exploits and breached data to file repositories like https://pastebin.com and code repositories like GitLab and GitHub.

Research Sources

Threat and vulnerability research sources include vendor websites, threat feeds, and vulnerability feeds from different providers, including security advisories from different providers. For example, the following link leads to the list of security advisories in Palo Alto products: https://security.paloaltonetworks.com. Likewise, the following link leads to the security advisories from Cisco: https://www.cisco.com/go/psirt. Most mature vendors disclose vulnerabilities affecting their products and services in a transparent way. Several commercial vendors also sell subscriptions to threat and vulnerability feeds.

You can learn and research new threats and vulnerabilities at security conferences, in academic journals, and from local industry groups. Examples of local industry groups include InfraGard (a collaborative effort between the FBI and the private sector), OWASP community chapters, DEF CON groups, and other local groups. Even ***social media*** (like Twitter) can be a source of security threat information.

You can also obtain deep technical information about threats and protocol deficiencies in the Internet Engineering Task Force (IETF) ***request for comments (RFC)*** drafts and publications (www.ietf.org).

The MITRE ATT&CK Framework

Threat hunters assume that an attacker has already compromised the network. Subsequently, they need to come up with a hypothesis of what is compromised and how an adversary could have performed the attack. For the threat hunting to be successful, hunters need to be aware of the ***adversary tactics***, ***techniques***, ***and procedures (TTPs)*** that modern attackers use. This is why many organizations use MITRE's ***ATT&CK*** framework to be able to learn about the tactics and techniques of adversaries. You will learn more about how MITRE's ATT&CK can be used in threat hunting.

Threat hunting is not a new concept. Many organizations have performed threat hunting for a long time. However, in the last decade many organizations have recognized that they either have to implement a threat-hunting program or enhance their existing program to better defend their

organization.

You can use the MITRE's ATT&CK framework to learn about the tactics and techniques that attackers used during their campaigns. The information in ATT&CK can be extremely useful for threat hunting.

ATT&CK (https://attack.mitre.org) is a collection of different matrices of tactics and techniques. The ATT&CK Matrix for Enterprise includes the tactics and techniques that adversaries use while preparing for an attack, including gathering of information (open-source intelligence, technical and people weakness identification, and more), and every single aspect of exploitation, post-exploitation, command and control, and other actions performed by the attacker. MITRE also provides a list of software (tools and malware) that adversaries use to carry out their attacks. You can access this list at https://attack.mitre.org/software.

You can also obtain detailed information about a specific tool or type of malware used by attackers for different purposes. For example, Figure 5-1 shows the well-known malicious memory-scraping and credential-dumping tool called Mimikatz in the MITRE ATT&CK Navigator.

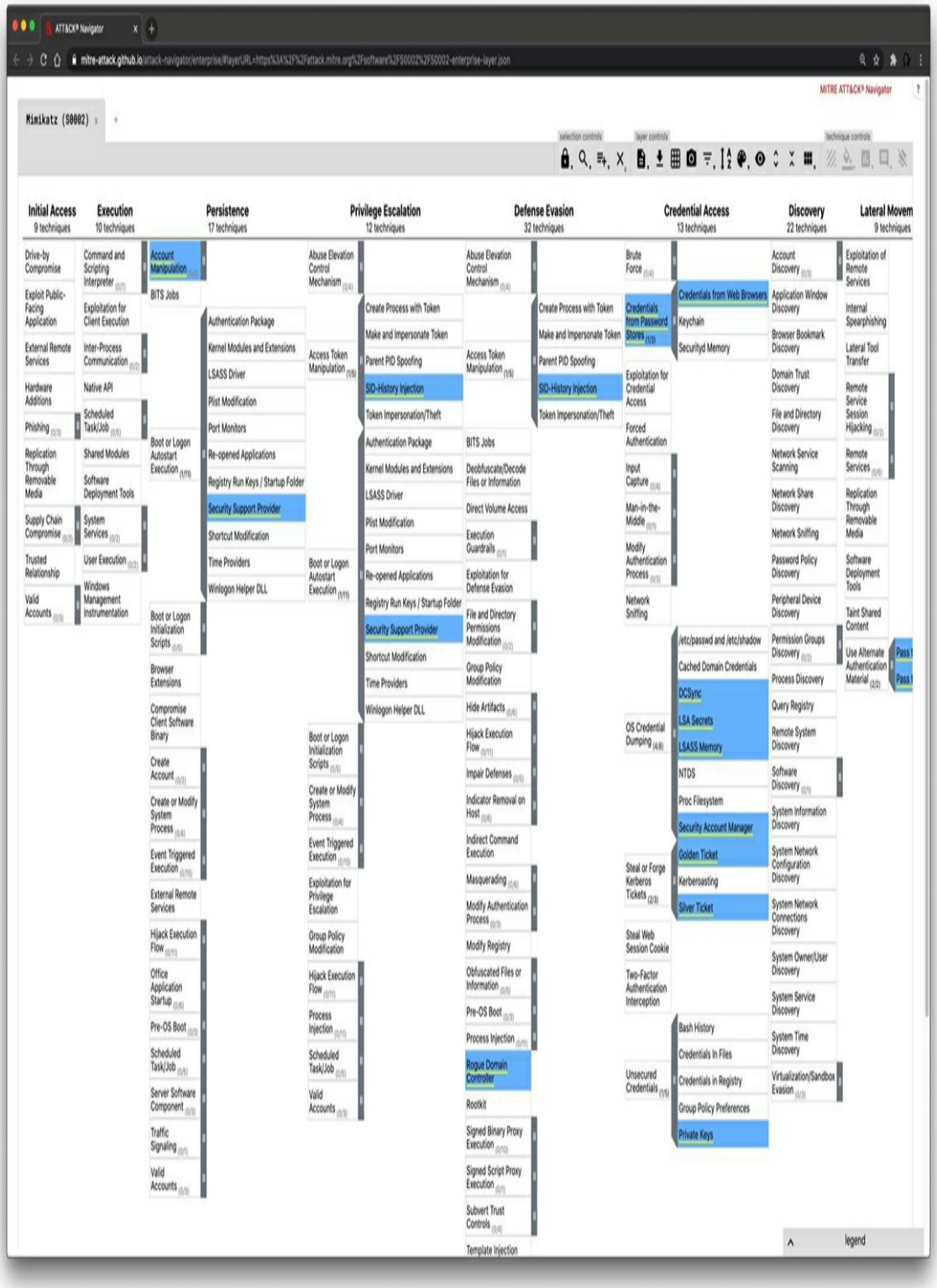


Figure 5-1 The MITRE ATT&CK Navigator

As illustrated in Figure 5-1, the MITRE ATT&CK Navigator shows all the different tactics and techniques where Mimikatz has been used in many different attacks by a multitude of attackers.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 5-2 lists a reference of these key topics and the page number on which each is found.

Table 5-2 Key Topics for Chapter 5

Key Topic Element	Description	Page Number
Section	Actors and Threats	
Paragraph	Understanding the attributes, intent, and motivations of threat actors	
Section	Attack Vectors	
Paragraph	Understanding threat intelligence and indicators of compromise	
Section	Structured Threat Information eXpression (STIX) and the Trusted Automated eXchange of Indicator Information (TAXII)	
Paragraph	Understanding predictive analysis	
Paragraph	Identifying threat and security information in file and code repositories	
Paragraph	Surveying threat and vulnerability research sources such as vendor websites, threat feeds, and vulnerability feeds	
Section	The MITRE ATT&CK Framework	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

script kiddie

hacktivist

criminal syndicates

organized crime

advanced persistent threat (APT)

state actors

shadow IT

authorized hacker

unauthorized hacker

semi-authorized hacker

intent

motivation

internal actor

external actor

threat intelligence

indicators of compromise

public information sharing centers

private information sharing centers

Information Sharing and Analysis Centers (ISACs)

open-source intelligence (OSINT)

dark web

vulnerability databases

automated indicator sharing (AIS)

Structured Threat Information eXpression (STIX)

Trusted Automated eXchange of Indicator Information (TAXII)

predictive analysis

threat maps

file and code repositories

social media

request for comments (RFC)

adversary tactics

techniques

and procedures (TTPs)

ATT&CK

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What includes the tactics and techniques that adversaries use while preparing for an attack, including gathering of information (open-source intelligence, technical and people weakness identification, and more)?
2. What standard was designed to document threat intelligence in a machine-readable format?
3. What standard was designed as a transport mechanism of threat

intelligence and to perform automated indicator sharing (AIS)?

4. What is the vulnerability database maintained by NIST?

5. What is the attack vector where evil twins are used?

6. You were hired to perform a penetration test against three different applications for a large enterprise. You are considered a(n) _____ hacker.

7. You are hired to investigate a cyber attack. Your customer provided different types of information collected from compromised systems such as malware hashes and the IP address of a potential command and control (C2). What are these elements often called?

8. What name is often used to describe a government or state-sponsored persistent and sophisticated attack?

9. What is the term used when an employee or a group of employees use IT systems, network devices, software, applications, and services without the approval of the corporate IT department?

Chapter 6. Understanding the Security Concerns Associated with Various Types of Vulnerabilities

This chapter covers the following topics related to Objective 1.6 (Explain the security concerns associated with various types of vulnerabilities) of the CompTIA Security+ SY0-601 certification exam:

- Cloud-based vs. on-premises vulnerabilities
- Zero-day
- Weak configurations
  - Open permissions
  - Unsecure root accounts
  - Errors
  - Weak encryption
  - Unsecure protocols
  - Default settings
  - Open ports and services
- Third-party risks
  - Vendor management
    - System integration
    - Lack of vendor support
  - Supply chain
  - Outsourced code development
  - Data storage

- Improper or weak patch management
    - Firmware
    - Operating system (OS)
    - Applications
- Legacy platforms
- Impacts
    - Data loss
    - Data breaches
    - Data exfiltration
    - Identity theft
    - Financial
    - Reputation
    - Availability loss

This chapter starts with an overview of cloud-based versus on-premises vulnerabilities. You then learn about zero-day vulnerabilities and related attacks. Weak configurations (such as open permissions, unsecure root accounts, errors, weak encryption, unsecure protocols, default settings, and open ports and services) can allow attackers to breach a network and compromise your systems. This chapter covers the details of those common weaknesses. In this chapter, you also learn about third-party risks, the consequences of improper or weak patch management, and the risk of running legacy platforms that are often vulnerable to attacks. The chapter concludes with an overview of the different types of impact categories when a cybersecurity attack or breach occurs.

“Do I Know This Already?” Quiz

The “Do I Know This Already?” quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the “Chapter Review Activities” section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire

chapter. Table 6-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 6-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Cloud-based vs. On-premises Vulnerabilities	1–2
Zero-day Vulnerabilities	3
Weak Configurations	4–5
Third-party Risks	6
Improper or Weak Patch Management	7
Legacy Platforms	8
The Impact of Cybersecurity Attacks and Breaches	9–10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which cloud service model offers computer networking, storage, load balancing, routing, and VM hosting?

a. IaaS

b. PaaS

c. SaaS

d. None of these answers are correct.

2. Which of the following is a security concern in cloud deployments?

a. Encryption

b. Authentication methods

c. Identity management

d. All of these answers are correct.

3. Which type of vulnerability is disclosed by an individual or exploited by an attacker before the creator of the software can create a patch to fix the underlying issue?

a. Cross-site scripting

b. Zero-day

c. Information disclosure

d. None of these answers are correct.

4. Which of the following can be considered a weak configuration that can allow attackers to perform an attack and compromise systems?

a. Default settings and passwords

b. Weak encryption

c. Open permissions

d. All of these answers are correct.

5. Which of the following is a weak protocol that should be avoided?

a. HTTP without encryption

b. Telnet

c. FTP without encryption

d. All of these answers are correct.

6. Which of the following should be considered when assessing third-party risks?

a. Vendor management

b. Supply chain

c. Outsourced code development

d. All of these answers are correct.

7. In Windows, what is a broadly released fix for a product-specific security-related vulnerability?

a. Security update

b. Service pack

c. Threat model

d. None of these answers are correct.

8. Which of the following is a disadvantage of running legacy platforms and products?

a. They are often affected by security vulnerabilities.

b. They do not have modern security features.

c. When a device is past the last day of support, vendors will not investigate or patch security vulnerabilities in those devices.

d. All of these answers are correct.

9. Which of the following could be categorized as different types of negative impact that a security breach could have in a corporation?

a. Financial

b. Reputation

c. Availability loss

d. All of these answers are correct.

10. Which of the following can be used by attackers to obfuscate their tactics when exfiltrating data from their target (victim)?

a. Encryption

b. Tunneling over a known protocol like DNS

c. Encoding

d. All of these answers are correct.

Foundation Topics

Cloud-based vs. On-premises Vulnerabilities

As time moves forward, more and more organizations are transferring some or all of their server and network resources to the cloud. Doing so creates many potential hazards and vulnerabilities that must be addressed by the security administrator and by the cloud provider. Top among these concerns are the servers, where all data is stored and accessed. Servers of all types should be hardened and protected from a variety of attacks in an effort to keep the integrity of data from being compromised. However, data must also be available. Consequently, you, as the security administrator, must strike a balance between security and availability. In this section, we discuss cloud-based threats as well as server vulnerabilities and how to combat them effectively.

Up until now we have focused on the individual computer system. Let's expand our security perimeter to include networks. Network design is extremely important in a secure computing environment. The elements that you include in your design can help to defend against many different types of network attacks. Being able to identify these network threats is the next step in securing your network. If you apply the strategies and defense mechanisms included in this chapter, you should be able to stave off most network-based assaults. Maintaining the security of the servers and network infrastructure of an organization is your job as the security administrator, but with the inclusion of the cloud, the areas of responsibility might vary. These responsibilities depend on how much of the cloud is provided by a third party and how much of the cloud is held privately within the organization's domain. Whether dealing with cloud providers, onsite cloud-based resources, locally owned servers and networks, or a mixture of all of them, you have a lot of duties and must understand not only security but how computer networking really functions. The CompTIA Security+ exam assumes that you

have a working knowledge of networks and that you have the CompTIA Network+ certification or commensurate experience. Hereafter, this book will work within that mindset and will refer directly to the security side of things as it progresses. So, put on your networking hat and let's begin with network design.

Historically, the term *cloud* was just a name for the Internet—anything beyond your network that you as a user couldn't see. Technically speaking, the cloud was the area of the telephone company's infrastructure; it was everything between one organization's demarcation point and the demarcation point of another organization. It included central offices, switching offices, telephone poles, circuit-switching devices, packet assemblers/disassemblers (PADs), packet-switching exchanges (PSEs), and so on. In fact, all these things, and much more, are still part of the cloud, in the technical sense. Back in the day, this term was used only by telecommunications professionals and network engineers.

Today, the term *cloud* has a somewhat different meaning. Almost everyone has heard of it and probably used it to some extent. It is used heavily in marketing, and the meaning is less technical and more service-oriented than it used to be. It takes the place of most intranets and extranets that had existed for decades before its emergence.

In ***on-premises*** environments, physical servers and virtual machines control the sending and receiving of all kinds of data over the network, including websites, email and text messaging, and data stored as single files and in database format. A great many of these "servers" are now virtual in the cloud, with more moving there every day. And the cloud, however an organization connects to it, is all about networking. Therefore, the cloud, virtualization, and servers in general are all thoroughly intertwined.

Cloud computing can be defined as a way of offering on-demand services that extend the capabilities of a person's computer or an organization's network. These might be free services, such as personal browser-based email from various providers, or they could be offered on a pay-per-use basis, such as services that offer data access, data storage, infrastructure, and online gaming. A network connection of some sort is required to make the

connection to the cloud and gain access to these services in real time.

Some of the benefits to an organization using cloud-based services include lowered cost, less administration and maintenance, more reliability, increased scalability, and possible increased performance. A basic example of a cloud-based service would be browser-based email. A small business with few employees definitely needs email, but it can't afford the costs of an email server and perhaps does not want to have its own hosted domain and the costs and work that go along with that. By connecting to a free web browser–based service, a small business can obtain near unlimited email, contacts, and calendar solutions. But, there is no administrative control, and there are some security concerns, which we discuss shortly.

Cloud computing services are generally broken down into several categories of services:

- ***Software as a service (SaaS)*:** This is the most commonly used and recognized of the three categories designed to provide a complete packaged solution. The software is rented out to the user. The service is usually provided through some type of front end or web portal. While the end user is free to use the service from anywhere, the company pays a per-use fee. Examples of SaaS offerings are Office 365, Gmail, Webex, Zoom, Dropbox, and Google Drive.

Note

Often compared to SaaS is the application service provider (ASP) model. SaaS typically offers a generalized service to many users. However, an ASP typically delivers a service (perhaps a single application) to a small number of users.

- ***Infrastructure as a service (IaaS)*:** This service offers computer networking, storage, load balancing, routing, and VM hosting. More and more organizations are seeing the benefits of offloading some of their networking infrastructure to the cloud.

- ***Platform as a service (PaaS)*:** This service provides various software solutions to organizations, especially the ability to develop applications in a virtual environment without the cost or administration of a physical platform. PaaS is used for easy-to-configure operating systems and on-demand computing. Often, this utilizes IaaS as well for an underlying infrastructure to the platform. Cloud-based virtual desktop environments (VDEs) and virtual desktop infrastructures (VDIs) are often considered to be part of this service but also can be part of IaaS.
- **Security as a service (SECaaS):** In this service a large service provider integrates its security services into the company's or customer's existing infrastructure. The concept is that the service provider can provide the security more efficiently and more cost effectively than a company can, especially if it has a limited IT staff or budget. The Cloud Security Alliance (CSA) defines various categories to help businesses implement and understand SECaaS, including encryption, data loss prevention (DLP), continuous monitoring, business continuity and disaster recovery (BCDR), vulnerability scanning, and much more.

Note

Periodically, new services will arrive, such as monitoring as a service (MaaS)—a framework that facilitates the deployment of monitoring within the cloud in a continuous fashion. There are many types of cloud-based services. If they don't fall into the preceding list, then they often fall under the category "anything as a service" (XaaS).

A cloud service provider (CSP) might offer one or more of these services. The National Institute of Standards and Technology (NIST) provides a great resource explaining the different cloud service models in Special Publication (SP) 800-145 (The NIST Definition of Cloud Computing). However, they are summarized here. There are different types of clouds used by organizations: public, private, hybrid, and community.

- ***Public cloud*:** In this type, a service provider offers applications and storage space to the general public over the Internet. Examples include

free, web-based email services and pay-as-you-go business-class services. The main benefits of this type of cloud include low (or zero) cost and scalability. Providers of public cloud space include Google Cloud Platform (GCP), Microsoft Azure, and Amazon Web Services (AWS).

- ***Private cloud*:** This type is designed for a particular organization in mind. As security administrator, you have more control over the data and infrastructure. A limited number of people have access to the cloud, and they are usually located behind a firewall of some sort in order to gain access to the private cloud. Resources might be provided by a third party or could come from the server room or data center.
- ***Hybrid cloud*:** This is a mixture of public and private clouds. Dedicated servers located within the organization and cloud servers from a third party are used together to form the collective network. In these hybrid scenarios, confidential data is usually kept in-house.
- ***Community cloud*:** This is another mix of public and private but one where multiple organizations can share the public portion. Community clouds appeal to organizations that usually have a common form of computing and storing of data.

The type of cloud an organization uses will be dictated by its budget, the level of security it requires, and the amount of manpower (or lack thereof) it has to administer its resources. While a private cloud can be very appealing, it is often beyond the ability of an organization, forcing that organization to seek the public or community-based cloud. However, it doesn't matter what type of cloud is used. Resources still have to be secured by someone, and you'll have a hand in that security one way or the other.

Cloud security hinges on the level of control you retain and the types of security controls you implement. When an organization makes a decision to use cloud computing, probably the most important security control concern to administrators is the loss of physical control of the organization's data. A more in-depth list of cloud computing security concerns includes lack of privacy, lack of accountability, improper authentication, lack of administrative control, data sensitivity and integrity problems, data segregation issues, location of data and data recovery problems, malicious insider attacks, bug exploitation, lack of investigative support when there is a

problem, and finally, questionable long-term viability. In general, everything that you worry about for your local network and computers! Keep in mind that cloud service providers can be abused as well: attackers often attempt to use providers' infrastructure to launch powerful attacks.

Solutions to these security issues include the following:

- **Complex passwords:** Strong passwords are beyond important; they are critical, as mentioned many times in this text. As of the writing of this book, accepted password schemes include the following:
  - **For general security:** A minimum of 10 characters, including at least one capital letter, one number, and one special character
  - **For confidential data:** A minimum of 15 characters, including a minimum of two each of capital letters, numbers, and special characters

  When it comes to the cloud, you might just opt to use the second option for every type of cloud. The reasoning is that public clouds can be insecure (you just don't know), and private clouds will most likely house the most confidential data. To enforce the type of password you want your users to choose, a strong server-based policy is recommended.

  However, passwords will not protect your data and systems. Passwords can be stolen and often are reused. This is why multifactor authentication is so important!

- **Powerful authentication methods:** Multifactor authentication can offer a certain amount of defense in depth. In this scenario, if one form of authentication is compromised, the other works as a backup. For example, in addition to a password, a person might be asked for biometric confirmation such as a thumbprint or voice authorization, for an additional PIN, or to swipe a smart card. Multifactor authentication may or may not be physically possible, depending on the cloud environment being used, but if at all possible, it should be considered. An example of a multifactor authentication solution is DUO (https://duo.com). It provides a way to integrate multifactor authentication with many different types of deployments, and users can use a phone app to confirm their identity and authenticate to a user or application.

- **Strong cloud data access policies:** We're talking the who, what, and when. When it comes to public clouds especially, you should specifically define which users have access, exactly which resources they have access to, and when they are allowed to access those resources. Configure strong passwords and consider two-factor authentication. Configure policies from servers that govern the users; for example, use Group Policy objects on a Windows Server domain controller. Audit any and all connected devices and apps. Consider storing different types of data with different services. Some services do better with media files, for example. Remember that cloud storage is not backup. Approach the backing up of data as a separate procedure.
- **Encryption:** Encryption of individual data files, whole disk encryption, digitally signed virtual machine files…the list goes on. Perhaps the most important is a robust public key infrastructure (PKI), which is discussed further in Chapter 25, "Implementing Public Key Infrastructure." The reason is that many users will access data through a web browser.
- **Standardization of programming:** The way applications are planned, designed, programmed, and run on the cloud should all be standardized from one platform to the next, and from one programmer to the next. Most important is standardized testing in the form of input validation, fuzzing, and known environment, unknown environment, or partially known environment testing.
- **Protection of all the data!:** This includes storage-area networks (SANs), general cloud storage, and the handling of big data (for example, astronomical data). When data is stored in multiple locations, it is easy for some to slip through the cracks. Detailed documentation of what is stored where (and how it is secured) should be kept and updated periodically. As a top-notch security administrator, you don't want your data to be tampered with. Therefore, implementing some cloud-based security controls can be very helpful. For example, consider the following: deterrent controls (prevent the tampering of data), preventive controls (increase the security strength of a system that houses data), corrective controls (reduce the effects of data tampering that has occurred), and detective controls (detect attacks in real time, and have a defense plan that can be immediately carried out).

What else are you, as administrator, trying to protect here? You're concerned with protecting the identity and privacy of your users (especially executives because they are high-profile targets). You need to secure the privacy of credit card numbers and other super-confidential information. You want to secure physical servers that are part of the server room or data center because they might be part of your private cloud. You desire protection of your applications with testing and acceptance procedures. (Keep in mind that these things all need to be done within contractual obligations with any third-party cloud providers.) And finally, you're interested in promoting the availability of your data. After all of your security controls and methods have been implemented, you might find that you have locked out more people than first intended. So, your design plan should contain details that will allow for available data, but in a secure manner.

Customers considering using cloud computing services should ask for transparency—or detailed information about the provider's security. The provider must be in compliance with the organization's security policies; otherwise, the data and software in the cloud become far less secure than the data and software within the customer's own network. This concept, and most of the concepts in the first half of this chapter, should be considered when planning whether to have data, systems, and infrastructure contained on-premises, in a hosted environment, on the cloud, or in a mix of those. If there is a mix of on-premises infrastructure and cloud-provider infrastructure, a company might consider a ***cloud access security broker (CASB)***—a software tool or service that acts as the gatekeeper between the two, allowing the company to extend the reach of its security policies beyond its internal infrastructure.

Other "Cloud"-based Concerns

Other technologies to watch out for are loosely connected with what can be called cloud technologies. One example is social media. Social media environments can include websites as well as special applications that are loaded directly on to the computer (mobile or desktop), among other ways to connect, both legitimate and illegitimate. People share the darndest things on social media websites, which can easily compromise the security of employees and data. The point? There are several ways to access social

media platforms, and it can be difficult for a security administrator to find every website, application, service, and port that is used by social media. In cases such as these, you might consider implementing more allow lists and block/deny lists to safeguard applications so that users are better locked down.

Another thing to watch for is P2P networks. File sharing, gaming, media streaming, and all the world is apparently available to a user—if the user knows where to look. However, P2P often comes with a price: malware and potential system infiltration. By the latter, I mean that computers can become unwilling participants in the sharing of data on a P2P network. This is one example in which the cloud invades client computers, often without the user's consent. Access to file sharing, P2P, and torrents also needs a permanent "padlock."

Then there's the dark web. Where Batman stores his data… No, the dark web is another type of P2P (often referred to as an F2F, meaning friends to friends) that creates connections between trusted peers (unlike most other P2Ps) but uses nonstandard ports and protocols. This makes it a bit more difficult to detect. The dark web is certainly the safe haven of illegal activities because they are designed specifically to resist surveillance. Computers that are part of an administrator's network, and more often, virtual machines in the admin's cloud, can be part of the dark web and can easily go undetected by the admin. In some cases, an employee of the organization (or an employee of the cloud provider) might have configured some cloud-based resources to join the dark web. This can have devastating legal consequences if illegal activities are traced to your organization. Thorough checks of cloud-based resources can help to prevent this situation. Also, screening of employees, careful inspection of service-level agreements with cloud providers, and the use of third-party IT auditors can avoid the possibility of dark web connectivity, P2P links, and improper use of social media.

Server Defense

Now we come down to it. Servers are the cornerstone of data. They store it, transfer it, archive it, and allow or disallow access to it. They need super-fast network connections that are monitored and baselined regularly. They require you to configure policies, check logs, and perform audits frequently. They

exist in networks both large and small, within public and private clouds, and are often present in virtual fashion. What it all comes down to is that servers contain the data and the services that everyone relies on. So, they are effectively the most important things to secure on your network.

Let’s break down five types of servers that are of great importance (in no particular order), and look at some of the threats and vulnerabilities to those servers, and ways to protect them.

File Servers

File server computers store, transfer, migrate, synchronize, and archive files. Really any computer can act as a file server of sorts, but examples of actual server software include Microsoft Windows Server and the various types of Linux server versions (for example, Ubuntu Server or Red Hat Server), not to mention UNIX. File servers are vulnerable to the same types of attacks and malware that typical desktop computers are. To secure file servers (and the rest of the servers on this list), you should employ hardening, updating, antimalware applications, software-based firewalls, hardware-based intrusion detection systems (HIDSs), and encryption, and be sure to monitor the server regularly.

Network Controllers

A network controller is a server that acts as a central repository of user accounts and computer accounts on the network. All users log in to this server. An example would be a Windows Server system that has been promoted to a domain controller (running Active Directory). In addition to the attacks mentioned for file servers, a domain controller can be the victim of LDAP injection. It also has Kerberos vulnerabilities, which can ultimately result in privilege escalation or spoofing. LDAP injection can be prevented with proper input validation. But in the specific case of a Windows domain controller, really the only way to keep it protected (aside from the preventive measures mentioned for file servers) is to install specific security software updates for the OS.

Email Servers

Email servers are part of the message server family. Here, a message server is

any server that deals with email, faxing, texting, chatting, and so on. For this section let's concentrate strictly on the email server. The most common of these is Microsoft Exchange. An Exchange Server might run POP3, SMTP, and IMAP, and allow for Outlook web-based connections. That's a lot of protocols and ports running. So, it's not surprising to hear some Exchange admins confess that running an email server can be difficult at times, particularly because it is vulnerable to XSS attacks, overflows, DoS/DDoS attacks, SMTP memory exploits, directory traversal attacks, and of course, spam. Bottom line: it has to be patched…a lot.

As an administrator, you need to keep on top of the latest vulnerabilities and attacks and possibly be prepared to shut down or quarantine an email server at a moment's notice. New attacks and exploits are constantly surfacing because email servers are a common and big target with a large attack surface. For spam, a hardware-based spam filter is most effective (such as one from Barracuda), but software-based filters can also help. To protect the integrity and confidentiality of email-based data, you should consider DLP and encryption. Security could also come in the form of secure POP3 and secure SMTP; for more about the specific secure email protocols, see Chapter 7, "Summarizing the Techniques Used in Security Assessments." Also, security can come as encrypted SSL/TLS connections, security posture assessment (SPA), secure VPNs, and other encryption types, especially for web-based connections. Web-based connections can be particularly insecure; great care must be taken to secure these connections. For example, push notification services for mobile devices are quite common. While TLS is normally used as a secure channel for the email connection, text and metadata can at times be sent as clear text. A solution to this is for the operating system to use a symmetrical key to encrypt the data payload. Vendors may or may not do this, so it is up to the email administrator to incorporate this added layer of security, or at least verify that push email providers are implementing it.

Thinking a little outside of the box, you could consider moving away from Microsoft (which is the victim of the most attacks) and toward a Linux solution such as the Java-based SMTP server built into Apache, or with a third-party tool such as Zimbra (or one of many others). These solutions are not foolproof, and still need to be updated, but it is a well-known fact that historically Linux has not been attacked as often as Microsoft (in general),

though the difference between the two in the number of attacks experienced has shrunk considerably since the turn of the millennium.

Web Servers

The web server could be the most commonly attacked server of them all. Examples of web servers include Microsoft's Internet Information Services (IIS), Apache HTTP Server (Linux), lighttpd (FreeBSD), Oracle iPlanet Web Server (Oracle), and iPlanet's predecessor Sun Java System Web Server (Sun Microsystems). Web servers in general can be the victim of DoS attacks, overflows, XSS and XSRF, remote code execution (RCE), and various attacks that make use of backdoors. For example, in IIS, if basic authentication is enabled, a backdoor could be created, and attackers could ultimately bypass access restrictions. An IIS administrator must keep up to date with the latest vulnerabilities by reading Microsoft Security Bulletins, such as this one that addresses possible information disclosure: https://technet.microsoft.com/library/security/ms12-073.

In general, as security administrator, you should keep up to date with ***Common Vulnerabilities and Exposures (CVE)*** as maintained by MITRE (https://cve.mitre.org/). The latest CVE listings for applications and operating systems can be found there and at several other websites.

Aside from the usual programmatic solutions to vulnerabilities such as XSS and standard updating and hot patching, you might consider adding and configuring a hardware-based firewall from Cisco, Fortinet, Palo Alto, or other similar company. Of course, HTTPS (be it SSL or, better yet, TLS) can be beneficial if the scenario calls for it. Once a server is secured, you can prove the relative security of the system to users by using an automated vulnerability scanning program (such as Netcraft) that leaves a little image on the web pages stating whether or not the site is secure and when it was scanned or audited.

Apache can be the casualty of many attacks as well, including privilege escalation, code injection, and exploits to the proxy portion of the software. PHP forms and the PHP engine could act as gateways to the Apache web server. Patches to known CVEs should be applied as soon as possible.

Note

You can find a list of CVEs to Apache HTTP Server (and the corresponding updates) at http://httpd.apache.org/security/.

When it comes to Apache web servers, you have to watch out for the web server attack called Darkleech. It takes the form of a malicious Apache module (specifically an injected HTML iframe tag within a PHP file). If loaded on a compromised Apache web server, it can initiate all kinds of attacks and deliver various payloads of malware and ransomware. Though Darkleech is not limited to Apache, the bulk of Darkleech-infected sites have been Apache-based.

Note

So much for Microsoft being less targeted than Linux. As time moves forward, it seems that no platform is safe. A word to the wise: don’t rely on any particular technology because of a reputation, and be sure to update and patch every technology you use.

As far as combatting Darkleech, a webmaster can attempt to query the system for PHP files stored in folders with suspiciously long hexadecimal names. If convenient for the organization, all iframes can be filtered out. Of course, the Apache server should be updated as soon as possible, and if necessary, taken offline while it is repaired. In many cases, this type of web server attack is very hard to detect, and sometimes the only recourse is to rebuild the server (or virtual server image) that hosts the Apache server.

Note

Another tool that some attackers use is archive.org. This website takes snapshots of many websites over time and stores them. They are accessible to anyone and can give attackers an idea of older (and possibly less secure) pages and scripts that used to run on a web server. It could be that these files and scripts are still

located on the web server even though they are no longer used. This is a vulnerability that you should be aware of. Strongly consider removing older unused files and scripts from web servers.

FTP Server

An FTP server can be used to provide basic file access publicly or privately. Examples of FTP servers include the FTP server built into IIS, the Apache FtpServer, and other third-party offerings such as FileZilla Server and Pure-FTPd.

The standard, default FTP server is pretty insecure. It uses well-known ports (20 and 21), doesn't use encryption by default, and has basic username/password authentication. As a result, FTP servers are often the victims of many types of attacks. Examples include bounce attacks (when a person attempts to hijack the FTP service to scan other computers), buffer overflow attempts (when an attacker tries to send an extremely long username or password or filename to trigger the overflow), and attacks on the anonymous account (if utilized).

If the files to be stored on the FTP server are at all confidential, you should consider additional security. You can do so by incorporating FTP software that utilizes secure file transfer protocols such as FTPS or SFTP. Additional security can be provided by using FTP software that uses dynamic assignment of data ports, instead of using port 20 every time. Encryption can prevent most attackers from reading the data files, even if they are able to get access to them. Of course, if not a public FTP, the anonymous account should be disabled.

There also are other, more sinister attacks lurking, ones that work in conjunction with the web server, which is often on the same computer, or part of the same software suite—for instance, the web shell. There are plenty of variants of the web shell, but here we describe its basic function. The web shell is a program that is installed on a web server by an attacker and is used to remotely access and reconfigure the server without the owner's consent.

Web shells are remote access Trojans (RATs) but are also referred to as backdoors because they offer an alternative way of accessing the website for

the attacker. The reason the web shell attack is described here in the FTP section is that it is usually the FTP server that contains the real vulnerability —weak passwords. Once an attacker figures out an administrator password of the FTP server (often through brute-force attempts), the attacker can easily install the web shell and effectively do anything desired to that web server (and/or the FTP server). It seems like a house of cards, and in a way, it is.

How can you prevent this from happening? First, you can increase the password security and change the passwords of all administrator accounts. Second, you can eliminate any unnecessary accounts, especially any superfluous admin accounts and the dreaded anonymous account. Next, you should strongly consider separating the FTP server and web server to two different computers or virtual machines. Finally, you should set up automated scans for web shell scripts (usually PHP files) or have the web server provider do so. If the provider doesn't offer that kind of scanning, you can use a different provider. If a web shell attack is accomplished successfully on a server, you must at the very least search for and delete the original RAT files and at worst reimage the system and restore from backup. This latter option is often necessary if the attacker has had some time to compromise the server. Some organizations have policies that state servers must be reimaged if they are compromised in any way, shape, or form. This solution is a way of starting anew with a clean slate, but it means a lot of configuring. But again, the overall concern here is the complexity of, and the frequency of changing, the password.

That's the short list of servers. There are plenty of others you need to be cognizant of, including DNS servers, application servers, virtualization servers, firewall/proxy servers, database servers, print servers, remote connectivity servers such as RRAS and VPN, and computer telephony integration (CTI) servers. If you are in charge of securing a server, be sure to examine the CVEs and bulletins for that software and be ready to hot-patch the system at a moment's notice. This means having an RDP, VNC, or other remote connection to that specific server ready to go on your desktop so that you can access it quickly.

Zero-day Vulnerabilities

A ***zero-day vulnerability*** is a type of vulnerability that is disclosed by an

individual or exploited by an attacker before the creator of the software can create a patch to fix the underlying issue. Attacks leveraging zero-day vulnerabilities can cause damage even after the creator knows of the vulnerability because it may take time to release a patch to prevent the attacks and fix damage caused by them.

Zero-day attacks can be prevented by using newer operating systems that have protection mechanisms and by updating those operating systems. They can also be prevented by using multiple layers of firewalls and by using application approved lists, which allow only known good applications to run. Collectively, these preventive methods are referred to as zero-day protection.

Table 6-2 summarizes programming vulnerabilities/attacks covered so far in this chapter.

Table 6-2 Programming Vulnerabilities and Attacks

Vulnerability	Description
Backdoor	An attack placed by programmers, knowingly or inadvertently, to bypass normal authentication, and other security mechanisms in place.
Buffer overflow	A vulnerability that occurs when a process stores data outside the memory that the developer intended.
Remote code execution (RCE)	A situation that occurs when an attacker obtains control of a target computer through some sort of vulnerability, gaining the power to execute commands on that remote computer.
Cross-site scripting (XSS)	A vulnerability that exploits the trust a user's browser has in a website through code injection, often in web forms.
Cross-site request forgery (XSRF)	A vulnerability that exploits the trust that a website has in a user's browser, which becomes compromised and transmits unauthorized commands to the website.
Code injection	An attack that occurs when user input in database web forms is not filtered correctly and is executed improperly. SQL injection is a very common example.
Directory traversal	A method of accessing unauthorized parent (or worse, root) directories.
Zero-day	A group of attacks executed on vulnerabilities in software before those vulnerabilities are known to the creator.

The CompTIA Security+ exam objectives don’t expect you to be a programmer, but they do expect you to have a basic knowledge of programming languages and methodologies so that you can help to secure applications effectively. I recommend a basic knowledge of programming languages used to build applications, such as Visual Basic, C++, C#, PowerShell, Java, and Python, as well as web-based programming languages such as HTML, JavaScript, and PHP, plus knowledge of database programming languages such as SQL. This foundation knowledge will help you not only on the exam but also when you, as security administrator, have to act as a liaison to the programming team or if you are actually involved in testing an application.

Weak Configurations

Weak configurations can be leveraged by attackers to compromise systems and networks. Often on-premises and cloud-based systems and applications could be deployed without security in mind. Sometimes, these weak configurations are due to lack of security awareness and understanding of common threats.

The following are some of the most prevalent types of weak configurations.

- **Open permissions:** User accounts can be added to individual computers or to networks. For example, a Windows client, Linux computer, or Mac can have multiple users. Larger networks that have a controlling server —for example, a Windows domain controller—enable user accounts that can access one or more computers on the domain. In some cases, users and/or applications are given access (permissions) to resources that they do not need to access (including files, folders, and systems). Applications should be coded and run in such a way as to maintain the principle of least privilege. Users should have access only to what they need. Processes should run with only the bare minimum access needed to complete their functions. However, this can be coupled with separation of privilege, where access to objects depends on more than one condition (for example, authentication plus an encrypted key).

- **Unsecure root accounts:** Out-of-the-box offerings should be as secure as possible. If possible, user password complexity and password aging default policies should be configured by the programmer, not the user. Administrative accounts (root accounts in Linux or administrator in Windows) should be protected at all times.
- **Errors:** Sometimes weak configurations are due to human errors that could lead to security nightmares. The potential for human error always exists.
- **Weak encryption:** Weak encryption or no encryption can be the worst thing that can happen to a wireless or wired network. This situation can occur for several reasons. For example, someone wants to connect an older device or a device that hasn't been updated, and that device can run only a weaker, older type of encryption. Weak cryptographic implementations can lead to sensitive data being exposed to attackers and could also have a direct impact to privacy. You will learn the basics of cryptographic concepts and encryption in Chapter 16, "Summarizing the Basics of Cryptographic Concepts."
- **Unsecure protocols:** Insecure protocols can also lead to the unauthorized exposure of sensitive data and can allow attackers to compromise systems and applications. Protocols such as Telnet, FTP (without encryption), Finger, SSL, old versions of SNMP, and others should be avoided at all times. Table 6-3 shows protocols and related ports, as well as the "more secure options" of some of those protocols.
- **Default settings:** A common adage in the security industry is, "Why do you need hackers if you have default settings and passwords?" Many organizations and individuals leave infrastructure devices such as routers, switches, wireless access points, and even firewalls configured with default passwords. Attackers can easily identify and access systems that use shared default passwords. It is extremely important to always change default manufacturer passwords and restrict network access to critical systems. A lot of manufacturers now require users to change the default passwords during initial setup, but some don't. Attackers can easily obtain default passwords and identify Internet-connected target systems. Passwords can be found in product documentation and compiled lists available on the Internet. Different examples are available

in the GitHub repository at https://github.com/The-Art-of-Hacking/h4cker), but dozens of other sites contain default passwords and configurations on the Internet. It is easy to identify devices that have default passwords and that are exposed to the Internet by using search engines such as Shodan (https://www.shodan.io).

- **Open ports and services:** Any unnecessary ports should be closed, and any open ports should be protected and monitored carefully. Although there are 1,024 well-known ports, for the exam you need to know only a handful of them, plus some that are beyond 1,024, as shown in Table 6-3. Remember that these inbound port numbers relate to the applications, services, and protocols that run on a computer, often a server. When it comes to the OSI reference model, the bulk of these protocols are application layer protocols. Examples of these protocols include HTTP, FTP, SMTP, SSH, DHCP, and POP3, and there are many more. Because these are known as application layer protocols, their associated ports are known as *application service ports*. The bulk of Table 6-3 is composed of application service ports. Some of the protocols listed make use of TCP transport layer connections only (for example, HTTP, port 80; and HTTPS, port 443). Some make use of UDP only (for example, SNMP, port 161). Many can use TCP *or* UDP transport mechanisms. Study Table 6-3 carefully now, bookmark it, and refer to it often.

Table 6-3 Ports and Their Associated Protocols

Port Number	Associated Protocol (or Keyword)	TCP/UDP Usage	Secure Version and Port	Usage
21	FTP	TCP	FTPS, port 989/990	Transfers files from host to host.
22	SSH	TCP or UDP		Secure Shell: Remotely administers network devices and systems. Also is used by Secure Copy (SCP) and Secure FTP (SFTP).
23	Telnet	TCP or UDP		Remotely administers network devices (deprecated).
25	SMTP	TCP	SMTP with SSL/TLS, port 465 or 587	Sends email.
49	TACACS+	TCP		Provides remote authentication. Can also use UDP, but TCP is the default. Compare with RADIUS.
53	DNS	TCP or UDP	DNSSEC	Resolves hostnames to IP addresses and vice versa.
69	TFTP	UDP		Is a basic version of FTP.
80	HTTP	TCP	HTTPS (uses SSL/TLS), port 443	Transmits web page data.

88	Kerberos	TCP or UDP		Provides network authentication; uses tickets.
110	POP3	TCP	POP3 with SSL/TLS, port 995	Receives email.
119	NNTP	TCP		Transports Usenet articles.
135	RPC/epmap/dcom-scm	TCP or UDP		Enables you to locate DCOM ports. Also known as RPC (Remote Procedure Call).
137–139	NetBIOS	TCP or UDP		Enables name querying, data sending, and NetBIOS connections.
143	IMAP	TCP	IMAP4 with SSL/TLS, port 993	Retrieves email, with advantages over POP3.
161	SNMP	UDP		Remotely monitors network devices.
162	SNMPTRAP	TCP or UDP		Sends Traps and InformRequests to the SNMP Manager on this port.
389	LDAP	TCP or UDP	LDAP over SSL/TLS, port 636	Maintains directories of users and other objects.

445	SMB	TCP		Provides shared access to files and other resources.
514	Syslog	UDP		Is used for computer message logging, especially for router and firewall logs. A secure version (Syslog over TLS) uses TCP as the transport mechanism and port 6514.
860	iSCSI	TCP		Is an IP-based protocol used for linking data storage facilities. Also uses port 3260 for the iSCSI target.
1433	Ms-sql-s	TCP		Opens queries to Microsoft SQL server.
1701	L2TP	UDP		Is a VPN protocol with no inherent security. Often used with IPsec.
1723	PPTP	TCP or UDP		Is a VPN protocol with built-in security.
1812/1813	RADIUS	UDP		Is an AAA protocol used for authentication (port 1812), authorization, and accounting (port 1813) of users. Also, ports 1645 and

				1646.
3225	FCIP	TCP or UDP		Encapsulates Fibre Channel frames within TCP/IP packets. Contrast with Fibre Channel over Ethernet (FCoE), which relies on the data link layer and doesn't rely on TCP/IP directly.
3389	RDP	TCP or UDP		Remotely views and controls other Windows systems.
3868	Diameter	TCP (or SCTP)		Is a protocol that can be used for authentication, authorization and accounting (AAA). Diameter is an alternative to the RADIUS protocol.

Note

You can find a complete list of ports and their corresponding protocols at http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Not all protocols have set port numbers. For example, the Real-Time Transport Protocol (RTP) and Secure RTP (SRTP) use a pair of

port numbers determined by the application that is streaming the audio and video information via RTP. They are selected from a broad range of ports (between 16384 and 32767).

Unnecessary applications and services use valuable hard drive space and processing power. More importantly, they can be vulnerabilities to an operating system. That's why many organizations implement the concept of *least functionality*. This means that an organization configures computers and other information systems to provide only the essential functions. Using this method, you restrict applications, services, ports, and protocols. This control —called CM-7—is described in more detail by NIST at https://nvd.nist.gov/800-53/Rev4/control/CM-7.

The United States Department of Defense describes this concept in DoD instruction 8551.01 at http://www.dtic.mil/whs/directives/corres/pdf/855101p.pdf.

It's this mindset that can help protect systems from threats that are aimed at insecure applications and services. For example, instant messaging programs can be dangerous. They might be fun for the user but usually are not productive in the workplace (to put it nicely); and from a security viewpoint, they often have backdoors that are easily accessible to attackers. Unless they are required by tech support, they should be discouraged and/or disallowed by rules and policies. You should be proactive when it comes to these types of programs. If a user can't install an IM program to a computer, then you will never have to remove it from that system. However, if you do have to remove an application like this, be sure to remove all traces that it ever existed. That is just one example of many, but it can be applied to most superfluous programs.

Other programs you should watch out for are remote control programs. Applications that enable remote control of a computer should be avoided if possible. For example, Remote Desktop Connection is a commonly used Windows-based remote control program. By default, this program uses inbound port 3389, which is well known to attackers—an obvious security threat. Consider using a different port if the program is necessary, and if not, make sure that the program's associated service is turned off and disabled. Check whether any related services need to be disabled as well. Then verify

that their inbound ports are no longer functional and that they are closed and secured. Confirm that any shares created by an application are disabled as well. Basically, remove all instances of the application or, if necessary, reimage the computer!

Uninstalling applications on a few computers is feasible, but what if you have a larger network? Say, one with 1,000 computers? You can't expect yourself or your computer techs to go to each and every computer locally and remove applications. That's when centrally administered management systems come into play. Examples include Microsoft's Endpoint Configuration Manager and the variety of mobile device management (MDM) suites available. These programs allow you to manage lots of computers' software, configurations, and policies, all from a local workstation.

Third-party Risks

When an organization is dealing with computer security, a *risk* is the possibility of a malicious attack or other threat causing damage or downtime to a computer system. Generally, this is done by exploiting vulnerabilities in a computer system or network. The more vulnerability, the more risk. Smart organizations are extremely interested in managing vulnerabilities and thereby managing risk. *Risk management* can be defined as the identification, assessment, and prioritization of risks, and the mitigating and monitoring of those risks. Specifically, in relation to computer hardware and software, risk management is also known as *information assurance (IA)*. Many well-known breaches have been caused by attackers leveraging misconfigured or vulnerable systems and applications in third-party vendors (such as a business partner or vendor).

Vendor management is crucial for any organization. Whether you are part of a small company or a global enterprise, third-party vendors have become essential to business operations. These third-party vendors include outsourcing companies, the software and hardware you purchase for your infrastructure, cloud service providers, and more. Risk management should include the evaluation of third-party vendors, along with their security

capabilities, and even include security requirements in contract negotiations. In addition, if you are employing contractors from an external vendor, you should always audit the level of access that those contractors have of your resources and data.

Many organizations hire external ***system integration*** companies to deploy new technologies and applications. You must understand your system's integrator security capabilities and understand what risks misconfigurations or vulnerable designs can bring to your organization.

Your organization may also leverage ***outsourced code development***. No software is immune to security vulnerabilities. You must have a way for the vendor to validate and test for any software security vulnerabilities that could be introduced in the outsourced code. You should ask vendors to demonstrate how they test for vulnerabilities including but not limited to static and dynamic analysis, as well as software composition analysis.

Lack of vendor support in current and legacy software and systems also introduces a big risk to your organization. If software and hardware vulnerabilities are not available due to lack of vendor support, those vulnerabilities can be leveraged by attackers to compromise your systems. The risk introduced in legacy platforms and software is covered later in this chapter. You should consistently reassess your vendor processes and support.

In Chapter 5, "Understanding Different Threat Actors, Vectors, and Intelligence Sources," you learned about ***supply chain*** attacks. Supply chain attacks are also called value-chain attacks. One of the most effective attacks for mass compromise is to attack the supply chain of a vendor to tamper with hardware and/or software. This tampering might occur in-house or, previously, while in transit through the manufacturing supply chain. In addition to traditional vendors of software and hardware, cloud offerings are also susceptible to supply chain attacks. Cloud services are an integral part of most businesses nowadays. The compromise of the underlying cloud infrastructure or services can be catastrophic. For example, an attacker

implanting a backdoor in all the base images used for your virtual machines in the cloud could lead to massive compromise of your applications and many of their customers.

Insecure ***data storage*** is a major concern of most companies nowadays—not only for data stored in your data center but also in the cloud. Without a doubt, the data needs to be encrypted, but more specifically three types of data: data in use, data at rest, and data in transit. *Data in processing* (also known as data in use) can be described as actively used data undergoing constant change; for example, it could be stored in databases or spreadsheets. *Data at rest* is inactive data that is archived—backed up to tape or otherwise. *Data in transit* (also known as data in motion) is data that crosses the network or data that currently resides in computer memory.

Redundant systems and data backups should also be taken seriously. The whole concept revolves around single points of failure. A *single point of failure* is an element, object, or part of a system that, if it fails, causes the whole system to fail. By implementing redundancy, you can bypass just about any single point of failure. You will learn more about data and system redundancy in Chapter 13, "Implementing Cybersecurity Resilience."

The computer file system used dictates a certain level of security. On Microsoft computers, the best option is to use NTFS, which is more secure, enables logging (oh so important), supports encryption, and has support for a much larger maximum partition size and larger file sizes. Just about the only place where FAT32 and NTFS are on a level playing field is that they support the same number of file formats. By far, then, NTFS is the best option. If a volume uses FAT or FAT32, it can be *converted* to NTFS using the following command:

```
convert volume /FS:NTFS
```

For example, if you want to convert a USB flash drive named M: to NTFS, the syntax would be

```
convert M: /FS:NTFS
```

There are additional options for the **convert** command. To see them, simply

type **convert /?** at the Command Prompt. NTFS enables file-level security and tracks permissions within access control lists (ACLs), which are a necessity in today's environment. Most systems today already use NTFS, but you never know about flash-based and other removable media. Using the **chkdsk** command at the Command Prompt or right-clicking the drive in the GUI and selecting Properties can tell you what type of file system it runs.

Generally, the best file system for Linux systems is ext4. It allows for the best and most configurable security. To find out the file system used by your version of Linux, use the **fdisk -l** command or **df -T** command.

System files and folders, by default, are hidden from view to protect a Windows system, but you never know. To permanently configure the system to not show hidden files and folders, navigate to the File Explorer (or Folder) Options dialog box. Then select the View tab, and under Hidden Files and Folders, select the Don't Show Hidden Files, Folders, or Drives radio button. To configure the system to hide protected system files, select the Hide Protected Operating System Files checkbox, located below the radio button previously mentioned. This way, you turn off the ability to view such files and folders as bootmgr and pagefile.sys. You might also need to secure a system by turning off file sharing, which in most versions of Windows can be done within the Network and Sharing Center.

In the past, I have made a bold statement: "Hard disks *will* fail." But it's all too true. It's not a matter of *if*; it's a matter of *when*. By maintaining and hardening the hard disk with various hard disk utilities, you attempt to stave off that dark day as long as possible. You can implement several strategies when maintaining and hardening a hard disk:

- **Remove temporary files:** Temporary files and older files can clog up a hard disk, cause a decrease in performance, and pose a security threat. It is recommended that Disk Cleanup or a similar program be used. Policies can be configured (or written) to run Disk Cleanup every day or at logoff for all the computers on the network.
- **Periodically check system files:** Every once in a while, it's a good idea to verify the integrity of operating system files. You can do a file

integrity check in the following ways:

- With the **chkdsk** command in Windows. This command examines the disk and provides a report. It can also fix some errors with the **/F** option.
- With the **SFC** (System File Checker) command in Windows. This utility checks and, if necessary, replaces protected system files. It can be used to fix problems in the operating system and in other applications such as Internet Explorer. A typical command you might type is **SFC /scannow**. You can use this command if **chkdsk** is not successful at making repairs.
- With the **fsck** command in Linux. This command is used to check and repair a Linux file system. The synopsis of the syntax is **fsck** [**-sAVRTNP**] [**-C** [**fd**]] [**-t fstype**] [**filesys ...**] [**--**] [**fs-specific-options**]. You can find more information about this command at the corresponding man page for **fsck**. A derivative, **e2fsck**, is used to check a Linux ext2fs (second extended file system). Also, you can download open-source data integrity tools for Linux, such as Tripwire.

- **Defragment drives:** Applications and files on hard drives become fragmented over time. For a server, this could be a disaster because the server cannot serve requests in a timely fashion if the drive is too thoroughly fragmented. To defragment the drive, you can use Microsoft's Disk Defragmenter, the command-line **defrag** command, or other third-party programs.
- **Back up data:** Backing up data is critical for a company. It is not enough to rely on a fault-tolerant array. Individual files or the entire system can be backed up to another set of hard drives, to optical discs, to tape, or to the cloud. Microsoft domain controllers' Active Directory databases are particularly susceptible to attack; the system state for these operating systems should be backed up in case the server fails and the Active Directory needs to be recovered in the future.
- **Use restoration techniques:** In Windows, restore points should be created on a regular basis for servers and workstations. The System Restore utility (rstrui.exe) can fix issues caused by defective hardware or

software by reverting back to an earlier time. Registry changes made by hardware or software are reversed in an attempt to force the computer to work the way it did previously. Restore points can be created manually and are also created automatically by the operating system before new applications, updates, or hardware are installed. macOS uses the Time Machine utility, which works in a similar manner. Though there is no similar tool in Linux, you can back up the ~/home directory to a separate partition. When these contents are decompressed to a new install, most of the Linux system and settings are restored. Another option in general is to use imaging (cloning) software. Remember that these techniques do not necessarily back up data and that you should treat the data as a separate entity that needs to be backed up regularly.

- **Consider whole disk encryption:** Finally, you can use whole disk encryption to secure the contents of the drive, making it harder for attackers to obtain and interpret its contents.

A recommendation I give to all my students and readers is to separate the operating system from the data physically. If you can have each on a separate hard drive, it can make things a bit easier just in case the OS is infected with malware (or otherwise fails). The hard drive that the OS inhabits can be completely wiped and reinstalled without worrying about data loss, and applications can always be reloaded. Of course, you should back up settings (or store them on the second drive). If a second drive isn't available, consider configuring the one hard drive as two partitions—one for the OS (or system) and one for the data. By doing this and keeping a well-maintained computer, you are effectively hardening the OS.

By maintaining the workstation or server, you are hardening it as well. This process can be broken down into six steps (and one optional step):

Step 1. Use a surge protector or UPS. Make sure the computer and other equipment connect to a surge protector, or better yet a UPS if you are concerned about power loss.

Step 2. Update the BIOS and/or UEFI. Flashing the BIOS isn't always necessary. Check the manufacturer's website for your motherboard to

see if an update is needed.

Step 3. Update the OS. For Windows, this step includes any combination hotfixes and any Windows Updates beyond that, and setting Windows to alert if there are any new updates. For Linux and macOS, it means simply updating the system to the latest version and installing individual patches as necessary.

Step 4. Update antimalware. This step includes making sure that there is a current license for the antimalware (antivirus and antispyware) and verifying that updates are turned on and the software is regularly scanning the system.

Step 5. Update the firewall. Be sure to have some kind of firewall installed and enabled; then update it. If it is Windows Defender Firewall, updates should happen automatically through Windows Updates. However, if you have a SOHO router with a built-in firewall or other firewall device, you need to update the device's ROM by downloading the latest image from the manufacturer's website.

Step 6. Maintain the disks. This instruction means running a disk cleanup program regularly and checking to see whether the hard disk needs to be defragmented from once a week to once a month depending on the amount of usage. It also means creating restore points, doing computer backups, or using third-party backup or drive imaging software.

Step 7. (Optional) Create an image of the system. After all your configurations and hardening of the OS are complete, you might consider creating an image of the system. Imaging the system is like taking a snapshot of the entire system partition. That information is saved as one large file, or a set of compressed files that can be saved anywhere. It's kind of like system restore but at another level. The beauty of this process is that you can reinstall the entire image if your system fails or is compromised, quickly and efficiently, with very little configuration necessary. You need to apply only the latest security and AV updates since the image was created. Of course, most imaging software has a price tag involved, but the investment can be well worth the cost if you are concerned about the time needed

to get your system back up and running in the event of a failure. This is the basis for standardized images in many organizations. By applying mandated security configurations, updates, and so on, and then taking an image of the system, you can create a snapshot in time that you can easily revert to if necessary, while being confident that a certain level of security is already embedded into the image.

Note

To clean out a system regularly, consider reimaging it, or if it's a mobile device, resetting it. Reimaging takes care of any pesky malware by deleting everything and reinstalling to the point in time of the image, or to factory condition. Although you will have to do some reconfigurations, the system will also run much faster because it has been completely cleaned out.

Improper or Weak Patch Management

Patch management is the planning, testing, implementing, and auditing of patches. You must adopt a proper patch management program to include all your *operating systems*, *firmware* of your devices in your infrastructure, and *applications* (on-premises and in the cloud).

To be considered secure, operating systems should have support for multilevel security and be able to meet government requirements. An operating system that meets these criteria is known as a ***trusted operating system (TOS)***. Examples of certified trusted operating systems include Windows 10, SELinux, FreeBSD (with the TrustedBSD extensions), and Red Hat Enterprise Server. For an OS to be considered a TOS, the manufacturer of the system must have strong policies concerning updates and patching.

Even without being a TOS, operating systems should be updated regularly. For example, Microsoft recognizes the deficiencies in an OS and possible exploits that could occur and therefore releases patches to increase OS

performance and protect the system.

Before you update, the first thing to do is to find out the version number, build number, and the patch level. For example, in Windows you can find out this information by opening the System Information tool (open the Run prompt and type **msinfo32.exe**). It is listed directly in the System Summary. You could also use the **winver** command. In addition, you can use the **systeminfo** command at the Command Prompt (a *great* information gatherer!) or simply the **ver** command.

Then you should check your organization's policies to see what level a system should be updated to. You should also test updates and patches on systems located on a clean, dedicated, testing network before going live with an update.

Windows uses the Windows Update program to manage updates to the system. Versions before Windows 10 include this feature in the Control Panel. It can also be accessed by typing **wuapp.exe** at the Run prompt. In Windows 10 it is located in the Settings section or can be opened by typing **ms-settings:windowsupdate** at the Run prompt.

There are various options for the installation of Windows updates, including Automatic Install, Defer Updates to a Future Date, Download with Option to Install, Manual Check for Updates, and Never Check for Updates. The types available to you will differ depending on the version of Windows. Businesses usually frown on Automatic Install, especially in the enterprise. Generally, as the security administrator, you want to specify what is updated, when it is updated, and to which computers it is updated. This way you eliminate problems such as patch level mismatch and network usage issues.

In some cases, your organization might opt to turn off Windows Update altogether. Depending on your version of Windows, turning it off may or may not be possible within the Windows Update program. However, your organization might go a step further and specify that the Windows Update *service* be stopped and disabled, thereby disabling updates. This can be done in the Services console window (Run > services.msc) or from the Command Prompt with one of the methods discussed earlier in the chapter; the service name for Windows Update is wuauserv.

Patches and Hotfixes

The best place to obtain patches and hotfixes is from the manufacturer's website. The terms *patches* and *hotfixes* are often used interchangeably. Windows updates are made up of hotfixes. Originally, a *hotfix* was defined as a single problem-fixing patch to an individual OS or application installed live while the system was up and running and without needing a reboot. However, the meaning of this term has changed over time and varies from vendor to vendor. (Vendors may even use both terms to describe the same thing.) For example, if you run the **systeminfo** command at the Command Prompt of a Windows computer, you see a list of hotfixes. They can be identified with the letters *KB* followed by seven numbers. Hotfixes can be single patches to individual applications, or they might affect the entire system.

On the other side of the spectrum, a gaming company might define hotfixes as a "hot" change to the server with no downtime, and no client download is necessary. The organization releases them if they are critical, instead of waiting for a full patch version. The gaming world commonly uses the terms *patch version, point release,* or *maintenance release* to describe a group of file updates to a particular gaming version. For example, a game might start at version 1 and later release an update known as 1.17. The .17 is the point release. (This could be any number, depending on the number of code rewrites.) Later, the game might release 1.32, in which .32 is the point release, again otherwise referred to as the patch version. This naming convention is common with other programs as well.

Administrators should keep up with software and hardware provider security advisories and disclosed vulnerabilities. This keeps you "in the know" when it comes to the latest updates. And this advice applies to server and client operating systems, server add-ons such as Microsoft Exchange or SQL Server, Office programs, web browsers, and the plethora of third-party programs that an organization might use. Your job just got a bit busier!

Of course, you are usually not concerned with updating games in the working world; you should remove them from a computer if they are found (unless perhaps you work for a gaming company). Multimedia software such as Camtasia, however, is prevalent in some companies, and web-based software such as a bulletin-board system is also common and susceptible to attack.

Patches generally carry the connotation of a small fix in the mind of the user or system administrator, so larger patches are often referred to as software updates, service packs, or something similar. However, if you were asked to fix a single security issue on a computer, a patch would be the solution you would want. For example, various Trojans attack older versions of Microsoft Office for Mac. To counter them, Microsoft released a specific patch for those versions of Office for Mac that disallow remote access by the Trojans.

Before installing an individual patch, you should determine whether it perhaps was already installed as part of a group update. For example, you might read that a particular version of macOS had a patch released for iTunes, and being an enthusiastic iTunes user, you might consider installing the patch. But you should first find out the version of the OS you are running; it might already include the patch. To find this information, simply click the Apple menu and then click About This Mac.

Remember: All systems need to be patched at some point. It doesn't matter if they are Windows, macOS, Linux, UNIX, Android, iOS, hardware appliances, kiosks, automotive computers…need I go on?

Unfortunately, sometimes patches are designed poorly, and although they might fix one problem, they could possibly create another, which is a form of software regression. Because you never know exactly what a patch to a system might do, or how it might react or interact with other systems, it is wise to incorporate patch management.

Patch Management

It is not wise to go running around the network randomly updating computers…not to say that you would do so, however! Patching, like any other process, should be managed properly. The process of patch management includes the planning, testing, implementing, and auditing of patches. I use the following four steps; other companies might have a slightly different patch management strategy, but each of the four concepts should be included:

- **Planning:** Before actually doing anything, you should set a plan into motion. The first thing that needs to be decided is whether the patch is necessary and whether it is compatible with other systems. The Microsoft Security Compliance Toolkit (SCT) is one example of a program that can identify security misconfigurations on the computers in your network, letting you know whether patching is needed. If the patch is deemed necessary, the plan should consist of a way to test the patch in a "clean" network on clean systems, how and when the patch will be implemented, and how the patch will be checked after it is installed.
- **Testing:** Before you automate the deployment of a patch among a thousand computers, it makes sense to test it on a single system or small group of systems first. These systems should be reserved for testing purposes only and should not be used by "civilians" or regular users on the network. I know, this is asking a lot, especially given the number of resources some companies have. But the more you can push for at least a single testing system that is not a part of the main network, the less you will be to blame if a failure occurs!
- **Implementing:** If the test is successful, the patch should be deployed to all the necessary systems. In many cases larger updates are done in the evening or over the weekend. Patches can be deployed automatically using software such as Microsoft's Endpoint Configuration Manager and third-party patch management tools.
- **Auditing:** When the implementation is complete, the systems (or at least a sample of systems) should be audited—first, to make sure the patch has taken hold properly, and second, to check for any changes or failures due to the patch. Microsoft's Endpoint Configuration Manager and third-party tools can be used in this endeavor.

Note

The concept of patch management, in combination with other application/OS hardening techniques, is collectively referred to as *configuration management*.

Some Linux-based and Mac-based programs and services also were

developed to help manage patching and the auditing of patches. Red Hat has services to help system administrators with all the RPMs they need to download and install, which can become a mountain of work quickly! And for those people who run GPL Linux, third-party services are also available. A network with a lot of mobile devices benefits greatly from the use of an MDM platform. Even with all these tools at an organization's disposal, sometimes patch management is just too much for one person, or for an entire IT department, and an organization might opt to contract that work out.

Several standards such as the *Common Security Advisory Framework (CSAF)* and the *Open Vulnerability and Assessment Language (OVAL)* are designed to provide a machine-readable format of security advisories to allow automated assessment of vulnerabilities. CSAF is also the name of the technical committee in the OASIS standards organization. CSAF is the successor of the *Common Vulnerability Reporting Framework (CVRF)*. CSAF enables different stakeholders across different organizations to share critical security-related information in a single format, speeding up information exchange and digestion.

OVAL is an international standard maintained by the Center for Internet Security (CIS). OVAL was originally created by MITRE and then transferred to CIS. You can obtain detailed information about the OVAL specification at https://oval.cisecurity.org. In short, OVAL can be defined in two parts: the OVAL Language and the OVAL Interpreter. OVAL is not a language like C++ but is an XML schema that defines and describes the XML documents to be created for use with OVAL.

OVAL has several uses, one of which is as a tool to standardize security advisory distributions. Software vendors need to publish vulnerabilities in a standard, machine-readable format. By including an authoring tool, definitions repository, and definition evaluator, OVAL enables users to regulate their security advisories. Other uses for OVAL include vulnerability assessment, patch management, auditing, threat indicators, and so on.

Legacy Platforms

You might be surprised at the number of small, medium, and large organizations that are still using ***legacy platforms*** and devices that have passed the vendor's last day of software and hardware support. These legacy devices often are core infrastructure devices such as routers and switches. If you run devices that have passed the last day that a vendor will provide software and hardware fixes, it is almost guaranteed that you will be running vulnerable devices because when devices are past the last day of support, vendors will not investigate or patch security vulnerabilities in those devices. This, in turn, introduces a huge risk to companies and service providers that run these devices.

The Impact of Cybersecurity Attacks and Breaches

It is obvious that major cybersecurity attacks and breaches can be catastrophic to many organizations. You just have to log in to your favorite news site or even Twitter to see the millions of dollars lost because of cybersecurity incidents and breaches.

The following are some of the most common categories of the impact of cybersecurity attacks and breaches:

- **Data loss, data breaches, and data exfiltration:** Data loss caused by a breach can range from just a one record to millions of records stolen by an attacker. Attackers can exfiltrate data from an organization using different techniques. They also can perform different types of obfuscation and evasion techniques to go undetected (including encoding of data, tunneling, and encryption). The impact on privacy because of a data breach can be very significant. Some of the data from a data breach can be "replaced" (such as a credit card number), but other data (like a Social Security number or health record) cannot.
- **Identity theft:** Criminals often use breached data for identity theft. They use or buy personal records and data from illegal sites in the dark web and other sources to steal the identity of individuals.
- **Financial:** The impact and consequences of a breach or a cybersecurity

incident can lead to fines and lawsuits against the company. In some cases, even executives could be fined or sued.

- **Reputation:** The brand and reputation of a company can also be damaged by major cybersecurity incidents and breaches. Customers can lose confidence and trust in the company that has lost their records because of a cybersecurity breach and in some cases because of negligence.
- **Availability loss:** Cybersecurity incidents can also lead to denial-of-service conditions that, in turn, can also cause significant financial impact. Outages from cybersecurity incidents could be catastrophic for companies and, in some cases, their customers.

When dealing with dollars, risk assessments should be based on a quantitative measurement of risk, impact, and asset value. This is why you have to build a good impact assessment methodology when you perform risk analysis. An excellent tool to create during risk assessment is a risk register, also known as a risk log, which helps to track issues and address problems as they occur. After the initial risk assessment, you will continue to use and refer to the risk register. It can be a great tool for just about any organization but can be of more value to certain types of organizations, such as manufacturers that utilize a supply chain. In this case, the organization would want to implement a specialized type of risk management called *supply chain risk management (SCRM)*. In this approach, the organization collaborates with suppliers and distributors to analyze and reduce risk.

When it comes to risk assessment, qualitative risk analysis is an assessment that assigns numeric values to the probability of a risk and the impact it can have on the system or network. Unlike its counterpart, quantitative risk assessment, it does not assign monetary values to assets or possible losses. It is the easier, quicker, and cheaper way to assess risk but cannot assign asset value or give a total for possible monetary loss. You will learn more about risk assessment in Chapter 34, "Summarizing Risk Management Processes and Concepts."

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 6-4 lists a reference of these key topics and the page number on which each is found.

Table 6-4 Key Topics for Chapter 6

Key Topic Element	Description	Page Number
Paragraph	Defining cloud computing	
List	Listing the cloud computing service models	
Table 6-2	Programming Vulnerabilities and Attacks	
List	Listing the most prevalent types of weak configurations	
Table 6-3	Ports and Their Associated Protocols	
Paragraph	Description of the role of vendor management and system management	
Paragraph	Description of outsourced code development and security concerns	
Paragraph	Description of the lack of vendor support that puts an organization at risk	
Paragraph	Description of data storage concerns	

List	Listing several strategies when maintaining and hardening a hard disk	
List	The process of maintaining the workstations and servers	
Paragraph	Description of security concerns with patch management	
List	Listing the patch management strategy phases	
Section	Legacy Platforms	
List	Surveying some of the most common categories of the impact of cybersecurity attacks and breaches	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

on-premises

cloud computing

Software as a service (SaaS)

infrastructure as a service (IaaS)

platform as a service (PaaS)

public cloud

private cloud

hybrid cloud

community cloud

cloud access security broker (CASB)

Common Vulnerabilities and Exposures (CVE)

zero-day vulnerability

vendor management

system integration

outsourced code development

lack of vendor support

supply chain

data storage

patch management

Trusted Operating System (TOS)

patches

legacy platforms

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What software tool or service acts as the gatekeeper between a cloud offering and the on-premises network, allowing an organization to extend the reach of its security policies beyond its internal infrastructure?

2. What is the name given to a type of vulnerability that is disclosed by an individual or exploited by an attacker before the creator of the software can create a patch to fix the underlying issue?

3. What cloud architecture model is a mix of public and private, but one where multiple organizations can share the public portion?

4. What type of vulnerability occurs when an attacker obtains control of a target computer through some sort of vulnerability, gaining the power to execute commands on that remote computer?

5. What protocol uses TCP ports 465 or 587 in most cases?

Chapter 7. Summarizing the Techniques Used in Security Assessments

This chapter covers the following topics related to Objective 1.7 (Summarize the techniques used in security assessments) of the CompTIA Security+ SY0-601 certification exam:

- Threat hunting
  - Intelligence fusion
  - Threat feeds
  - Advisories and bulletins
  - Maneuver
- Vulnerability scans
  - False positives
  - False negatives
  - Log reviews
  - Credentialed vs. non-credentialed
  - Intrusive vs. non-intrusive
  - Application
  - Web application
  - Network
  - Common Vulnerabilities and Exposures (CVE) and Common Vulnerability Scoring System (CVSS)
  - Configuration review
- Syslog/Security information and event management (SIEM)

    - Review reports
    - Packet capture
    - Data inputs
    - User behavior analysis
    - Sentiment analysis
    - Security monitoring
    - Log aggregation
    - Log collectors
- Security orchestration, automation, and response (SOAR)

This chapter starts by introducing threat hunting and how the threat-hunting process leverages threat intelligence. Then you learn about vulnerability management tasks, such as keeping up with security advisories and performing vulnerability scans. You also learn about the importance of collecting logs (such as system logs [syslogs]) and analyzing those logs in a Security Information and Event Management (SIEM) system. In addition, you learn how security tools and solutions have evolved to provide Security Orchestration, Automation, and Response (SOAR) capabilities to better defend your network, your users, and your organizations overall.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 7-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 7-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Threat Hunting	1–3
Vulnerability Scans	4–6
Syslog and Security Information and Event Management (SIEM)	7–8
Security Orchestration, Automation, and Response (SOAR)	9–10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. What is the act of proactively and iteratively looking for threats in your organization that may have bypassed your security controls and monitoring capabilities?

a. Threat intelligence

b. Threat hunting

c. Threat binding

d. None of these answers are correct.

2. Which of the following provides a matrix of adversary tactics, techniques, and procedures that modern attackers use?

a. ATT&CK

b. CVSS

c. CVE

d. All of these answers are correct.

3. Which identifier is assigned to disclosed vulnerabilities?

a. CVE

b. CVSS

c. ATT&CK

d. TTP

4. Which broad term describes a situation in which a security device triggers an alarm, but no malicious activity or actual attack is taking place?

a. False negative

b. True negative

c. False positive

d. True positive

5. Which of the following is a successful identification of a security attack or a malicious event?

a. True positive

b. True negative

c. False positive

d. False negative

6. Which of the following occurs when a vulnerability scanner logs in to the targeted system to perform deep analysis of the operating system, running applications, and security misconfigurations?

a. Credentialed scan

b. Application scan

c. Noncredentialed scan

d. None of these answers are correct.

7. Which of the following are functions of a SIEM?

a. Log collection

b. Log normalization

c. Log correlation

d. All of these answers are correct.

8. Which solution allows security analysts to collect network traffic metadata?

a. NetFlow

b. SIEM

c. SOAR

d. None of these answers are correct.

9. Which solution provides capabilities that extend beyond traditional SIEMs?

a. SOAR

b. CVSS

c. CVE

d. IPFIX

10. Which of the following can be capabilities and benefits of a SOAR solution?

a. Automated vulnerability assessment

b. SOC playbooks and runbook automation

c. Orchestration of multiple SOC tools

d. All of these answers are correct.

Foundation Topics

Threat Hunting

No security product or technology in the world can detect and block all

security threats in the continuously evolving threat landscape (regardless of the vendor or how expensive it is). This is why many organizations are tasking senior analysts in their computer security incident response team (CSIRT) and their security operations center (SOC) to hunt for threats that may have bypassed any security controls that are in place. This is why threat hunting exists.

Threat hunting is the act of proactively and iteratively looking for threats in your organization. This chapter covers details about threat-hunting practices, the operational challenges of a threat-hunting program, and the benefits of a threat-hunting program.

The threat-hunting process requires deep knowledge of the network and often is performed by SOC analysts (otherwise known as investigators, threat hunters, tier 2 or tier 3 analysts, and so on). Figure 7-1 illustrates the traditional SOC tiers and where threat hunters typically reside. In some organizations (especially small organizations), threat hunting could be done by anyone in the SOC because the organization may not have a lot of resources (analysts). The success of threat hunting completely depends on the maturity of the organization and the resources available.

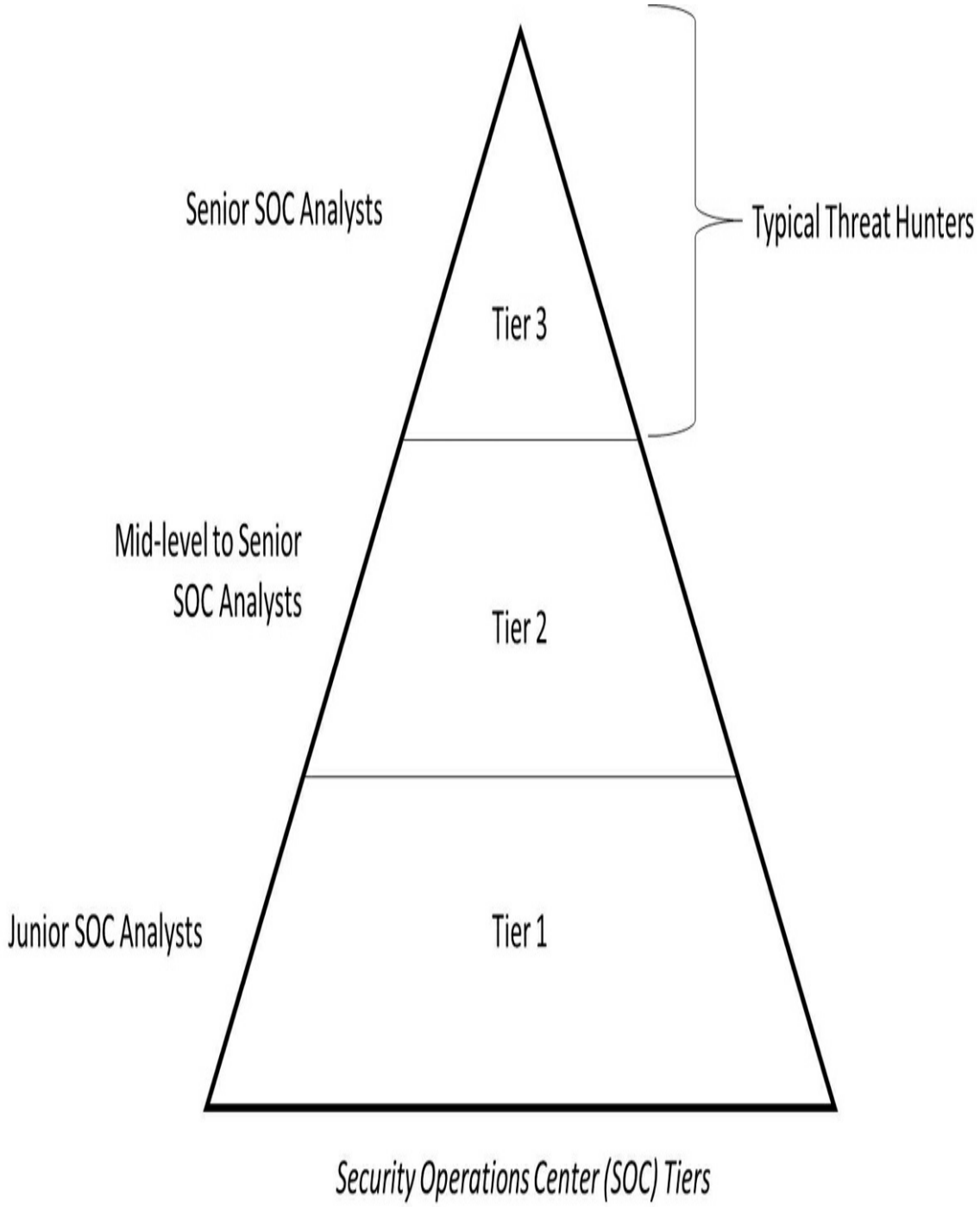


Figure 7-1 The SOC Tiers

Some organizations might have a dedicated team within or outside the SOC to perform threat hunting. However, one of the common practices is to have the hunters embedded within the SOC.

Threat hunters assume that an attacker has already compromised the network. Subsequently, they need to come up with a hypothesis of what is compromised and how an adversary could have performed the attack. For the threat hunting to be successful, hunters need to be aware of the adversary tactics, techniques, and procedures (TTPs) that modern attackers use. This is why many organizations use MITRE's ATT&CK framework to be able to learn about the tactics and techniques of adversaries. Later in this chapter you learn more about how MITRE's ATT&CK can be used in threat hunting.

Threat hunting is not a new concept. Many organizations have performed threat hunting for a long time. However, in the last decade many organizations have adopted new ways to enhance the threat-hunting process with automation and orchestration.

Threat hunting is not the same as the traditional SOC incident response (reactive) activities. Threat hunting is also not the same as vulnerability management (the process of patching vulnerabilities across the systems and network of your organization, including cloud-based applications in some cases). However, some of the same tools and capabilities may be shared among threat hunters, SOC analysts, and vulnerability management teams. Tools and other capabilities such as data analytics, TTPs, vulnerability feeds, and ***threat feeds*** may be used across the different teams and analysts in an organization.

A high-level threat-hunting process includes the following steps:

Step 1. Threat hunting starts with a trigger based on an anomaly, threat intelligence, or a hypothesis (what could an attacker have done to the organization?). From that moment you should ask yourself: "Do we really need to perform this threat-hunting activity?" or "What is the scope?"

Step 2. Then you identify the necessary tools and methodologies to conduct the hunt.

Step 3. Once the tools and methodologies are identified, you reveal new attack patterns, TTPs, and so on.

Step 4. You refine your hunting tactics and enrich them using data analytics. Steps 2–3 can take one cycle or be iterative and involve multiple loops (depending on what you find and what additional data and

research need to be done).

Step 5. A successful outcome could be that you identify and mitigate the threat. However, you need to recognize that in some cases this may not be the case. You may not have the necessary tools and capabilities, or there was no actual threat. This is why the success of your hunting program depends on the maturity of your capabilities and organization as a whole.

You can measure the maturity of your threat-hunting program within your organization in many ways. Figure 7-2 shows a matrix that can be used to evaluate the maturity level of your organization against different high-level threat-hunting elements.

THREAT HUNTING MATURITY LEVEL

THREAT HUNTING HIGH-LEVEL ELEMENTS	Initial (Minimal) Level 1	Intermediate Level 2	Innovative and Leading Level 3
Threat Intelligence and Data Collection	Limited access of threat intelligence and collection of data	High collection of certain types of threat intelligence and data	High collection of many types of threat intelligence and data
Hypothesis Creation	Responds only to existing SIEM, IPS/IDS, firewalls logs, etc.	Combines traditional logs with TTPs and threat intelligence	Combines traditional logs with TTPs and threat intelligence and develop automated threat risk scoring
Tools and Techniques for Hunting Hypothesis Testing	Reactive alerts and SIEM searches.	Simple tools and analytics leveraging some visualizations, but mostly a manual effort.	Advanced search capabilities, visualizations, creating new tools and not depending on traditional tools.
TTP Detection	None, only traditional SIEM reactive detection	Identification of indicators of compromise (IoCs) and new attack trends.	Able to detect adversary TTPs, IoCs, and create automation for the SOC to routinely detect them in the future.
Analytics and Automation	None	Limited analytics and automation	Create automated tools for the SOC to routinely detect threats in the future.

Figure 7-2 Threat-Hunting Maturity

These threat-hunting maturity levels can be categorized as easily as level 1, 2, and 3, or more complex measures can be used.

When it comes to threat intelligence and threat hunting, automation is key! Many organizations are trying to create threat ***intelligence fusion*** techniques to automatically extract threat intelligence data from heterogeneous sources to analyze such data. The goal is for the threat hunter and network defender to maneuver quickly—and faster than the attacker. This way, you can stay

one step ahead of threat actors and be able to mitigate the attack.

Security Advisories and Bulletins

In Chapter 5, "Understanding Different Threat Actors, Vectors, and Intelligence Sources," you learned how vendors, coordination centers, security researchers, and others publish ***security advisories*** and bulletins to disclose vulnerabilities. Most of the vulnerabilities disclosed in the public are assigned ***Common Vulnerability and Exposure (CVE)*** identifiers. CVE is a standard created by MITRE (www.mitre.org) that provides a mechanism to assign an identifier to vulnerabilities so that you can correlate the reports of those vulnerabilities among sites, tools, and feeds.

> **Note**
>
> You can obtain additional information about CVE at https://cve.mitre.org.

One of the most comprehensive and widely used vulnerability databases is the National Vulnerability Database (NVD) maintained by the National Institute of Standards and Technology (NIST). NVD provides information about vulnerabilities disclosed worldwide.

> **Note**
>
> You can access the NVD and the respective vulnerability feeds at https://nvd.nist.gov.

Most mature vendors such as Microsoft, Intel, and Cisco publish security advisories and bulletins in their websites and are CVE Numbering Authorities (CNAs). CNAs can assign CVEs to disclosed vulnerabilities and submit the information to MITRE and subsequently to NVD.

> **Note**
>
> You can find additional information about CNAs at https://cve.mitre.org/cve/cna.html.

The following links include examples of security advisories and bulletins published by different vendors:

- **Cisco:** https://www.cisco.com/go/psirt
- **Microsoft:** https://www.microsoft.com/en-us/msrc
- **Red Hat:** https://access.redhat.com/security/security-updates
- **Palo Alto:** https://security.paloaltonetworks.com

Vulnerability disclosures in security advisories are often coordinated among multiple vendors. Most of the products and applications developed nowadays use open-source software. Vulnerabilities in open-source software could affect hundreds or thousands of products and applications in the industry. In addition, vulnerabilities in protocols such as TLS, TCP, BGP, OSPF, and WPA could also affect numerous products and software. Patching open-source and protocol-related vulnerabilities among upstream and downstream vendors is not an easy task and requires good coordination. Figure 7-3 shows the high-level process of a coordinated vulnerability disclosure and underlying patching.

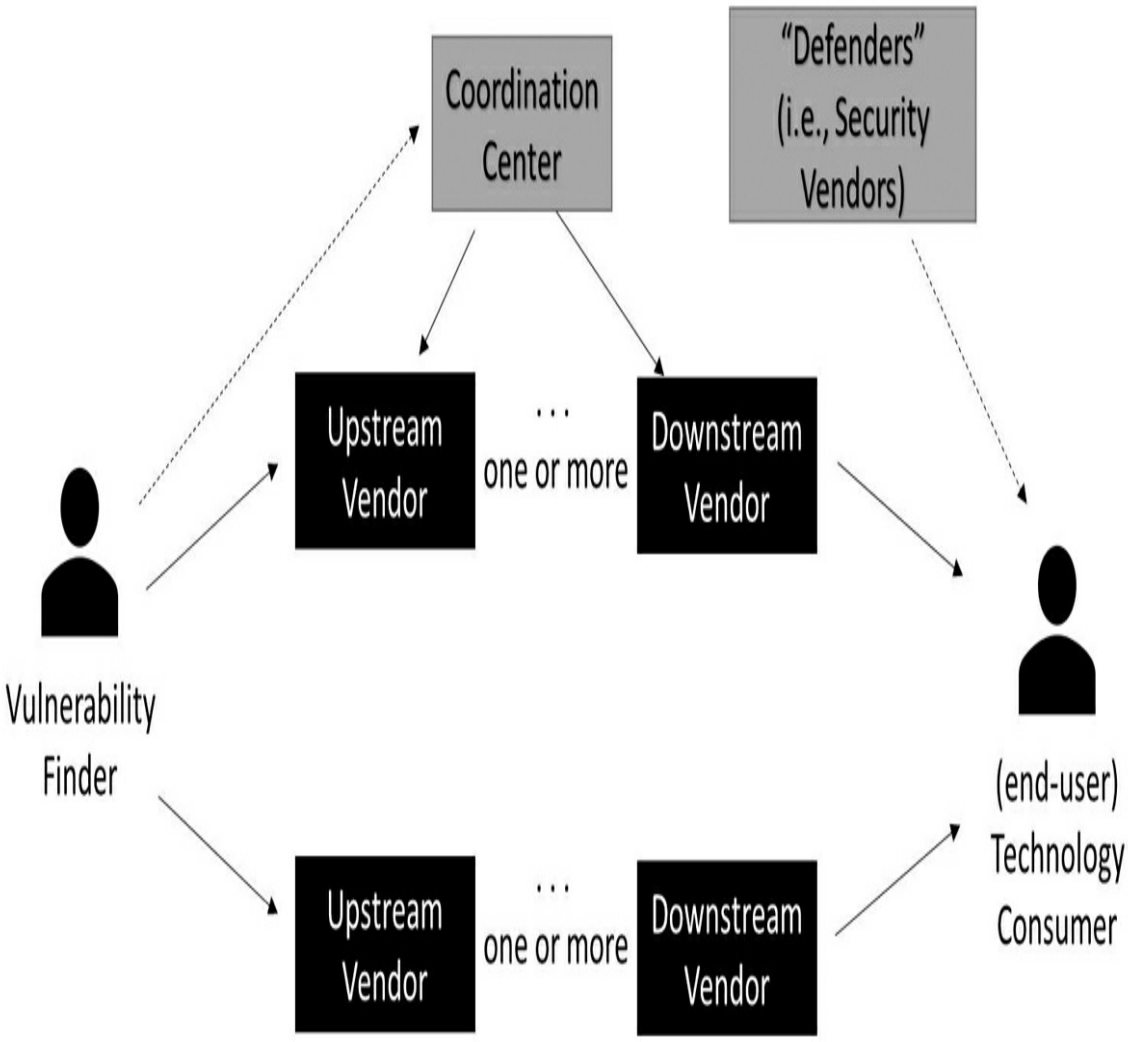


Figure 7-3 Coordinated Vulnerability Disclosures

The following steps are illustrated in Figure 7-3:

1. The finder (this can be anyone—a security researcher, customer, security company, an internal employee of a vendor) finds a security vulnerability and reports it to a vendor. The finder can also contact a vulnerability coordination center (such as www.cert.org) to help with the coordination and disclosure.
2. The upstream vendors triage and patch the vulnerability.
3. There could be one or more downstream vendors that also need to patch the vulnerability. In some cases, the coordination center may also interact with downstream vendors in the notification.

4. Security vendors (such as antivirus/antimalware, intrusion detection, and prevention technology providers) may obtain information about the vulnerability and create signatures or any other capabilities to help the end user detect and mitigate an attack caused by the vulnerability.
5. The end user is notified of the patch and the vulnerability.

Tip

The preceding process can take days, weeks, months, or even years! Although this process looks very simple in an illustration like the one in Figure 7-3, it is very complicated in practice. For this reason, the Forum of Incident Response and Security Teams (FIRST) has created a Multi-Party Coordination and Disclosure special interest group (SIG) to help address these challenges. You can obtain details about guidelines and practices for multiparty vulnerability coordination and disclosure at https://www.first.org/global/sigs/vulnerability-coordination/multiparty/.

Vulnerability Scans

Vulnerability management teams often use other tools such as vulnerability scanners and software composition analysis (SCA) tools. Figure 7-4 illustrates how a typical automated vulnerability scanner works.

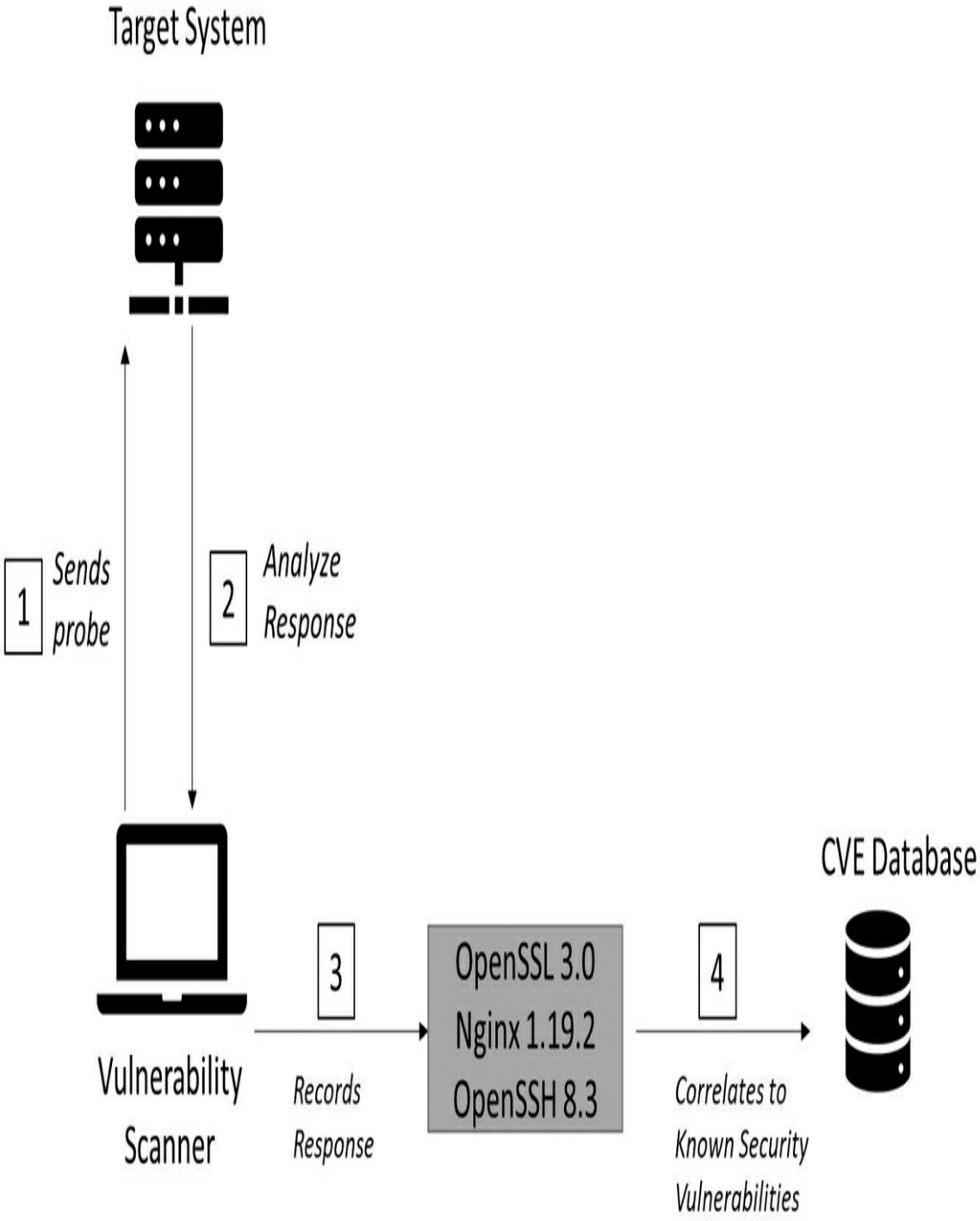


Figure 7-4 Coordinated Vulnerability Disclosures

The following are the steps illustrated in Figure 7-4. Keep in mind that vulnerability scanners are all different, but most follow a process like this:

1. In the discovery phase, the scanner uses a tool such as Nmap to perform host and port enumeration. Using the results of the host and port enumeration, the scanner begins to probe open ports for more information.
2. When the scanner has enough information about the open port to determine what software and version are running on that port, it records that information in a database for further analysis. The scanner can use various methods to make this determination, including banner information.
3. The scanner tries to determine if the software that is listening on the target system is susceptible to any known vulnerabilities. It does this by correlating a database of known vulnerabilities against the information recorded in the database about the target services.
4. The scanner produces a report on what it suspects could be vulnerable. Keep in mind that these results are often false positive and need to be validated.

One of the main challenges with automated vulnerability scanners is the number of false positives and false negatives. ***False positive*** is a broad term that describes a situation in which a security device triggers an alarm, but no malicious activity or actual attack is taking place. In other words, false positives are false alarms, and they are also called benign triggers. False positives are problematic because by triggering unjustified alerts, they diminish the value and urgency of real alerts. Having too many false positives to investigate becomes an operational nightmare, and you most definitely will overlook real security events.

There are also ***false negatives***, which is the term used to describe a network intrusion device's inability to detect true security events under certain circumstances—in other words, a malicious activity that is not detected by the security device.

A ***true positive*** is a successful identification of a security attack or a malicious event. A ***true negative*** occurs when the intrusion detection device identifies an activity as acceptable behavior and the activity is actually acceptable.

There are also different types of vulnerability scanners:

- ***Application scanners*:** Used to assess application-specific vulnerabilities and operate at the upper layers of the OSI model
- ***Web application scanners*:** Used to assess web applications and web services (such as APIs)
- ***Network and port scanners*:** Used to determine what TCP or UDP ports are open on the target system

Credentialed vs. Noncredentialed

To reduce the number of false positives, some vulnerability scanners have the capability to log in to a system to perform additional tests and see what programs, applications, and open-source software may be running on a targeted system. These scanners can also ***review logs*** on the target system. They can also perform ***configuration reviews*** to determine if a system may be configured in an unsecure way.

Intrusive vs. Nonintrusive

Vulnerability scanners sometimes can send numerous IP packets at a very fast pace (***intrusive***) to the target system. These IP packets can potentially cause negative effects and even crash the application or system. Some scanners can be configured in such a way that you can throttle the probes and IP packets that it sends to the target system in order to be ***nonintrusive*** and to not cause any negative effects in the system.

Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (or ***CVSS***) is an industry standard used to convey information about the severity of vulnerabilities. In

CVSS, a vulnerability is evaluated under three aspects, and a score is assigned to each of them. These three aspects (or groups) are the base, temporal, and environmental groups.

- The ***base group*** represents the intrinsic characteristics of a vulnerability that are constant over time and do not depend on a user-specific environment. This is the most important information and the only mandatory information to obtain for a vulnerability score.
- The ***temporal group*** assesses the vulnerability as it changes over time.
- The ***environmental group*** represents the characteristic of a vulnerability taking into account the organization's environment.

The CVSS score is obtained by taking into account the base, temporal, and environmental group information. The score for the base group is between 0 and 10, where 0 is the least severe and 10 is assigned to highly critical vulnerabilities (for example, for vulnerabilities that could allow an attacker to remotely compromise a system and get full control). Additionally, the score comes in the form of a vector string that identifies each of the components used to make up the score. The formula used to obtain the score takes into account various characteristics of the vulnerability and how the attacker is able to leverage these characteristics. CVSS defines several characteristics for the base, temporal, and environmental groups.

Tip

You can read and refer to the latest CVSS specification documentation, examples of scored vulnerabilities, and a calculator at www.first.org/cvss.

The base group defines exploitability metrics that measure how the vulnerability can be exploited, and impact metrics that measure the impact on confidentiality, integrity, and availability. In addition to these two, a metric called scope change (S) is used to convey the impact on systems that are affected by the vulnerability but do not contain vulnerable code.

Exploitability metrics include the following:

- **Attack Vector (AV):** Represents the level of access an attacker needs to

have to exploit a vulnerability. It can assume four values:

- Network (N)
- Adjacent (A)
- Local (L)
- Physical (P)

- **Attack Complexity (AC):** Represents the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. The values can be one of the following:
  - Low (L)
  - High (H)
- **Privileges Required (PR):** Represents the level of privileges an attacker must have to exploit the vulnerability. The values are as follows:
  - None (N)
  - Low (L)
  - High (H)
- **User Interaction (UI):** Captures whether user interaction is needed to perform an attack. The values are as follows:
  - None (N)
  - Required (R)
- **Scope (S):** Captures the impact on systems other than the system being scored. The values are as follows:
  - Unchanged (U)
  - Changed (C)

The Impact metrics include the following:

- **Confidentiality Impact (C):** Measures the degree of impact to the confidentiality of the system. It can assume the following values:
  - Low (L)
  - Medium (M)

    - High (H)
- **Integrity Impact (I):** Measures the degree of impact to the integrity of the system. It can assume the following values:
    - Low (L)
    - Medium (M)
    - High (H)
- **Availability Impact (A):** Measures the degree of impact to the availability of the system. It can assume the following values:
    - Low (L)
    - Medium (M)
    - High (H)

The temporal group includes three metrics:

- **Exploit code maturity (E):** Measures whether or not public exploits are available
- **Remediation Level (RL):** Indicates whether a fix or workaround is available
- **Report Confidence (RC):** Indicates the degree of confidence in the existence of the vulnerability

The environmental group includes two main metrics:

- **Security Requirements (CR, IR, AR):** Indicate the importance of confidentiality, integrity, and availability requirements for the system
- **Modified Base Metrics (MAV, MAC, MAPR, MUI, MS, MC, MI, MA):** Allow the organization to tweak the base metrics based on specific characteristics of the environment

For example, a vulnerability that could allow a remote attacker to crash the system by sending crafted IP packets would have the following values for the base metrics:

- Access Vector (AV) would be Network because the attacker can be anywhere and can send packets remotely.

- Attack Complexity (AC) would be Low because it is trivial to generate malformed IP packets.
- Privilege Required (PR) would be None because no privileges are required by the attacker on the target system.
- User Interaction (UI) would also be None because the attacker does not need to interact with any user of the system in order to carry out the attack.
- Scope (S) would be Unchanged if the attack does not cause other systems to fail.
- Confidentiality Impact (C) would be None because the primary impact is on the availability of the system.
- Integrity Impact (I) would be None because the primary impact is on the availability of the system.
- Availability Impact (A) would be High because the device becomes completely unavailable while crashing and reloading.

CVSS also defines a mapping between a CVSS Base Score quantitative value and a qualitative score. Table 7-2 provides the qualitative-to-quantitative score mapping.

Table 7-2 Qualitative-to-Quantitative Score Mapping

Rating	CVSS Base Score
None	0.0
Low	0.1–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

Tip

Organizations can use the CVSS score as input to their own risk management processes to evaluate the risk related to a vulnerability and then prioritize the vulnerability remediation.

Syslog and Security Information and Event Management (SIEM)

Security Information and Event Management (SIEM) is a specialized device or software used for ***security monitoring***; it collects, correlates, and helps security analysts analyze logs from multiple systems. SIEM typically allows for the following functions:

- **Log collection:** This includes receiving information from devices with multiple protocols and formats, storing the logs, and providing historical reporting and log filtering. A ***log collector*** is software that is able to receive logs from multiple sources (***data input***) and in some cases offers storage capabilities and log analysis functionality.
- **Log normalization:** This function extracts relevant attributes from logs received in different formats and stores them in a common data model or template. This allows for faster event classification and operations. Non-normalized logs are usually kept for archive, historical, and forensic purposes.
- ***Log aggregation*****:** This function aggregates information based on common information and reduces duplicates.
- **Log correlation:** This is probably one of the most important SIEM function. It refers to the ability of the system to associate events gathered by various systems, in different formats and at different times, and create a single actionable event for the security analyst or investigator. Often the quality of SIEM is related to the quality of its correlation engine.
- **Reporting:** Event visibility is also a key functionality of SIEM. Reporting capabilities usually include real-time monitoring and

historical base reports.

Most modern SIEMs also integrate with other information systems to gather additional contextual information to feed the correlation engine. For example, they can integrate with an identity management system to get contextual information about users or with NetFlow collectors to get additional flow-based information.

Note

NetFlow is a technology created by Cisco to collect network metadata about all the different "flows" of traffic on your network. There's also the Internet Protocol Flow Information Export (***IPFIX***), which is a network flow standard led by the Internet Engineering Task Force (IETF). IPFIX was designed to create a common, universal standard of export for flow information from routers, switches, firewalls, and other infrastructure devices. IPFIX defines how flow information should be formatted and transferred from an exporter to a collector. IPFIX is documented in RFC 7011 through RFC 7015 and RFC 5103. Cisco NetFlow Version 9 is the basis and main point of reference for IPFIX. IPFIX changes some of the terminologies of NetFlow, but in essence they are the same principles of NetFlow Version 9.

Several commercial SIEM systems are available. Here's a list of some commercial SIEM solutions:

- Micro Focus ArcSight
- LogRhythm
- IBM QRadar
- Splunk

Figure 7-5 shows how SIEM can collect and process logs from routers, network switches, firewalls, intrusion detection, and other security products that may be in your infrastructure. It can also collect and process logs from applications, antivirus, antimalware, and other host-based security solutions.

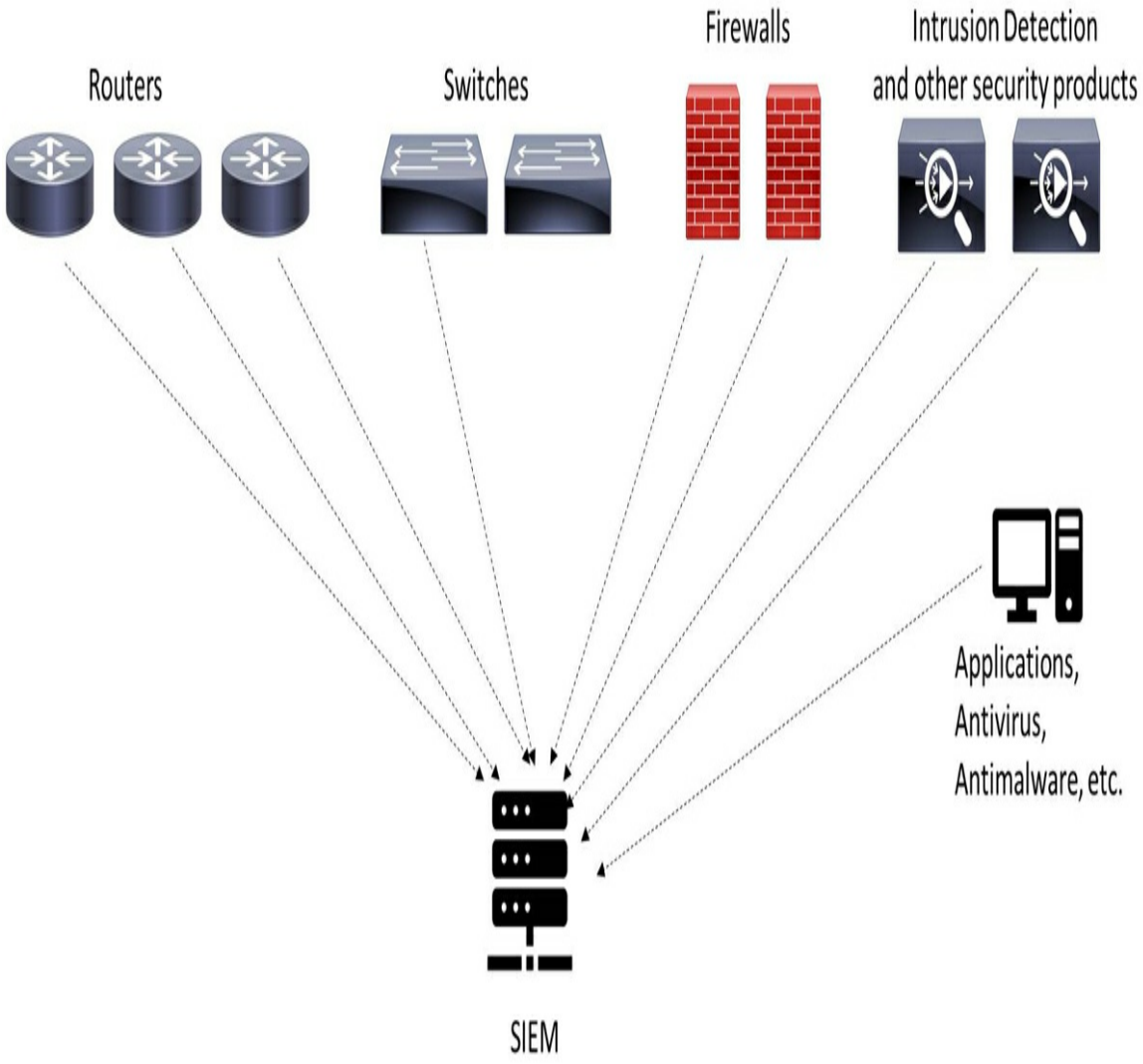


Figure 7-5 SIEM Collecting and Processing Logs from Disparate Systems

Security operation center analysts and security engineers often collect ***packet captures*** during the investigation of a security incident. Packet captures provide the greatest detail about each transaction happening in the network. Full packet capture has been used for digital forensics for many years. However, most malware and attackers use encryption to be able to bypass and obfuscate their transactions. IP packet metadata can still be used to potentially detect an attack and determine the attacker's tactics and techniques.

One of the drawbacks of collecting full packet captures in every corner of

your network is the requirement for storage because packet captures in busy networks can take a significant amount of disk space. This is why numerous organizations often collect network metadata with NetFlow or IPFIX and store such data longer than when collecting packet captures.

Several sophisticated security tools also provide ***user behavior analysis*** mechanisms in order to potentially find insiders (internal attackers). Similarly, they provide insights of user behavior even if they do not present a security threat.

Organizations can also deploy ***sentiment analysis*** tools and solutions to help monitor customer sentiment and brand reputation. Often these tools can also reveal the intent and tone behind social media posts, as well as keep track of positive or negative opinions. Threat actors can also try to damage a company's reputation by creating fake accounts and bots in social media platforms like Twitter, Facebook, or Instagram. Attackers can use these fake accounts and bots to provide negative public comments against the targeted organization.

Security Orchestration, Automation, and Response (SOAR)

CSIRT analysts typically work in an SOC utilizing many tools to monitor events from numerous systems (firewalls, applications, IPS, DLP, endpoint security solutions, and so on). Typically, these logs are aggregated in a SIEM. Modern SOCs also use ***Security Orchestration***, ***Automation***, ***and Response (SOAR)*** systems that extend beyond traditional SIEMs.

The tools in the SOC are evolving and so are the methodologies. For example, now security analysts not only respond to basic cyber events but also perform threat hunting in their organizations. SOAR is a set of solutions and integrations designed to allow organizations to collect security threat data and alerts from multiple sources. SOAR platforms take the response capabilities of SIEM to the next level. SOAR solutions supplement, rather than replace, the SIEM. They allow the cybersecurity team to extend its reach

by automating the routine work of cybersecurity operations.

> **Tip**
>
> Unlike traditional SIEM platforms, SOAR solutions can also be used for threat and vulnerability management, security incident response, and security operations automation.

Deploying SOAR and SIEM together in solutions makes the life of SOC analysts easier. SOAR platforms accelerate incident response detection and eradication times because they can automatically communicate information collected by SIEM with other security tools. Several traditional SIEM vendors are changing their products to offer hybrid SOAR/SIEM functionality.

Another term adopted in the cybersecurity industry is Extended Detection and Response (XDR). XDR is a series of systems working together that collects and correlates data across hosts, mobile devices, servers, cloud workloads, email messages, web content, and networks, enabling visibility and context into advanced threats. The goal of an XDR system is to allow security analysts to analyze, prioritize, hunt, and remediate cybersecurity threats to prevent data loss and security breaches.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 7-3 lists a reference of these key topics and the page number on which each is found.

Table 7-3 Key Topics for Chapter 7

Key Topic Element	Description	Page Number
Paragraph	Defining threat hunting	
Paragraph	Understanding security advisories, bulletins, and what a CVE is	
Paragraph	Understanding false positives and false negatives	
Paragraph	Comparing credentialed vs. noncredentialed vulnerability scanners	
Paragraph	Comparing intrusive and nonintrusive vulnerability scanners	
Paragraph	Defining what SIEM is	
Paragraph	Understanding the SOAR concept	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

threat hunting

threat feeds

intelligence fusion

security advisories

Common Vulnerability and Exposures (CVE)

false positive

false negatives

true positive

true negative

application scanners

web application scanners

network and port scanners

review logs

configuration reviews

intrusive

nonintrusive

CVSS

base group

temporal group

environmental group

Security Information and Event Management (SIEM)

security monitoring

log collector

data input

Log aggregation

IPFIX

packet captures

user behavior analysis

sentiment analysis

Security Orchestration

Automation

and Response (SOAR)

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What type of vulnerability scanner can be used to assess vulnerable web services?
2. What documents do vendors, vulnerability coordination centers, and security researchers publish to disclose security vulnerabilities?
3. What term is used to describe an organization that can assign CVEs to vulnerabilities?
4. What public database can anyone use to obtain information about security vulnerabilities affecting software and hardware products?
5. How many score “groups” are supported in CVSS?
6. A vulnerability with a CVSS score of 4.9 is considered a _____ severity vulnerability.
7. What is the process of iteratively looking for threats that may have bypassed your security controls?

Chapter 8. Understanding the Techniques Used in Penetration Testing

This chapter covers the following topics related to Objective 1.8 (Explain the techniques used in penetration testing) of the CompTIA Security+ SY0-601 certification exam:

- Penetration testing
  - Known environment
  - Unknown environment
  - Partially known environment
  - Rules of engagement
  - Lateral movement
  - Privilege escalation
  - Persistence
  - Cleanup
  - Bug bounty
  - Pivoting
- Passive and active reconnaissance
  - Drones
  - War flying
  - War driving
  - Footprinting
  - OSINT

- Exercise types
    - Red-team
    - Blue-team
    - White-team
    - Purple-team

Penetration testing (otherwise known as ethical hacking) has been extremely popular in the last several years. A penetration tester is someone who mimics what an attacker can do to an organization when finding and exploiting security vulnerabilities. In this chapter, you learn the details about penetration testing methodologies, rules of engagement, and how security researchers also participate in bug bounties. You also learn how penetration testers (pen testers) perform passive and active reconnaissance and the different types of security exercises.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 8-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 8-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Penetration Testing	1–4
Passive and Active Reconnaissance	5–7
Exercise Types	8–10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

---

1. Which term is used to define the practice of mimicking a threat actor by using the same methodologies and tools to find and exploit vulnerabilities with the permission of the system or network owner?

a. Ethical hacking

b. Pen testing

c. Penetration testing

d. All of these answers are correct.

2. Which of the following is a type of penetration testing where the tester starts out with a significant amount of information about the organization and its infrastructure?

a. Known environment

b. Unknown environment

c. Partially known environment

d. None of these answers are correct.

3. Which of the following are elements of the penetration pre-engagement phase?

a. Developing the rules of engagement document

b. Negotiating contracts

c. Creating the statement of work (SOW)

d. All of these answers are correct.

4. Which of the following elements are typically included in the rules of engagement document during a penetration testing?

a. Testing timeline

b. Location of the testing

c. The security controls that could potentially detect or prevent testing

d. All of these answers are correct.

5. Which term is used when a penetration tester uses public records to perform passive reconnaissance?

a. OSINT gathering

b. Scanning

c. Banner fingerprinting

d. Shodan

6. Which tool can be used to perform active reconnaissance?

a. Nmap

b. Nessus

c. Nikto

d. All of these answers are correct.

7. Which term is used to describe when attackers or pen testers fly drones to perform reconnaissance of a location or eavesdrop wireless networks?

a. War driving

b. War flying

c. Wireless flying

d. None of these answers are correct.

8. Which of the following is an example of a blue team?

a. CSIRT

b. Pen testing teams

c. Offensive security teams

d. None of these answers are correct.

9. What term is used to describe how an organization integrates the defensive capabilities of a blue team with the adversarial techniques used by the red team?

a. Advanced red teaming

b. Adversarial emulation

c. Purple teaming

d. None of these answers are correct.

10. Which term is often used to define the team that focuses in security governance, regulatory compliance, and risk management?

a. White team

b. Purple team

c. Red team

d. Blue team

Foundation Topics

Penetration Testing

A penetration tester is typically also referred to as an ethical hacker. The term *ethical hacker* describes a person who acts as an attacker and evaluates the security posture of a computer network for the purpose of minimizing risk. The term *hacker* has been used in many different ways and has many different definitions. Most people in a computer technology field would consider themselves hackers by the simple fact that they like to tinker. This is obviously not a malicious thing. The key factor here in defining ethical versus nonethical hacking is that the latter involves malicious intent. A security researcher looking for vulnerabilities in products, applications, or web services is considered an ethical hacker if he or she responsibly discloses those vulnerabilities to the vendors or owners of the targeted research. However, the same type of “research” performed by someone who then uses the same vulnerability to gain unauthorized access to a target network/system

would be considered a nonethical hacker. We could even go so far as to say that someone who finds a vulnerability and discloses it publicly without working with a vendor is considered a nonethical hacker—because this could lead to the compromise of networks/systems by others who use this information in a malicious way.

The truth is that, as an ethical hacker, you use the same tools to find vulnerabilities and exploit targets as do nonethical hackers. However, as an ethical hacker, you would typically report your findings to the vendor or customer you are helping to make more secure. You would also try to avoid performing any tests or exploits that might be destructive in nature. An ethical hacker's goal is to analyze the security posture of a network's or system's infrastructure in an effort to identify and possibly exploit any security weaknesses found and then determine if a compromise is possible. This process is called security ***penetration testing*** or ethical hacking.

So, why do we need penetration testing? Well, first of all, as someone who is responsible for securing and defending a network/system, you want to find any possible paths of compromise before the bad guys do. For years we have developed and implemented many different defensive techniques (for instance, antivirus programs, firewalls, intrusion prevention systems [IPSs], antimalware). We have deployed defense-in-depth as a method to secure and defend our networks. But how do we know if those defenses really work and whether they are enough to keep out the bad guys? How valuable is the data that we are protecting, and are we protecting the right things? These are some of the questions that should be answered by a penetration test.

If you build a fence around your yard with the intent of keeping your dog from getting out, maybe it only needs to be 4 feet tall. However, if your concern is not the dog getting out but an intruder getting in, then you need a different fence—one that would need to be much taller than 4 feet. Depending on what you are protecting, you might also want razor wire on the top of the fence to deter the bad guys even more.

When it comes to information security, you need to do the same type of assessments on your networks and systems. You need to determine what you are protecting and whether your defenses can hold up to the threats that are imposed on them. This is where penetration testing comes in. Simply implementing a firewall, an IPS, antimalware, a virtual private network, a

web application firewall (WAF), and other modern security defenses isn't enough. You also need to test their validity. And you need to do this on a regular basis. As you know, networks and systems change constantly. This means the attack surface can change as well, and when it does, you need to consider reevaluating the security posture by way of a penetration test.

When talking about penetration testing methods, you are likely to hear the terms ***known environment***, ***unknown environment***, or ***partially known environment*** testing. These terms are used to describe the perspective from which the testing is performed, as well as the amount of information that is provided to the tester:

- **Known environment:** In this environment the tester starts out with a significant amount of information about the organization and its infrastructure. The tester would normally be provided things like network diagrams, IP addresses, configurations, and a set of user credentials. If the scope includes an application assessment, the tester might also be provided the source code of the target application. The idea of this type of test is to identify as many security holes as possible. In an unknown environment test, the scope may be only to identify a path into the organization and stop there. With "known environment" testing, the scope would typically be much broader and include internal network configuration auditing and scanning of desktop computers for defects. Time and money are typically deciding factors in the determination of which type of penetration test to complete. If a company has specific concerns about an application, a server, or a segment of the infrastructure, it can provide information about that specific target to decrease the scope and the amount of time spent on the test but still uncover the desired results. With the sophistication and capabilities of adversaries today, it is likely that most networks will be compromised at some point, and a known environment approach is not a bad option.
- **Unknown environment:** In this environment the tester is typically provided only a very limited amount of information. For instance, the tester may be provided only the domain names and IP addresses that are

in scope for a particular target. The idea of this type of limitation is to have the tester start out with the perspective that an external attacker might have. Typically, an attacker would first determine a target and then begin to gather information about the target, using public information and gaining more and more information to use in attacks. The tester would not have prior knowledge of the target's organization and infrastructure. Another aspect of unknown environment testing is that sometimes the network support personnel of the target may not be given information about exactly when the test is taking place. This type of environment allows for a defense exercise to take place as well, and it also eliminates the issue of a target preparing for the test and not giving a real-world view of how the security posture really looks.

- **Partially known environment:** This type of penetration test is somewhat of a hybrid approach between the preceding two types. The penetration testers may be provided credentials but not full documentation of the network infrastructure. This way, the testers could still provide results of their testing from the perspective of an external attacker's point of view. Considering the fact that most compromises start at the client and work their way throughout the network, a good approach would be a scope where the testers start on the inside of the network and have access to a client machine. Then they could pivot throughout the network to determine what the impact of a compromise would be.

A number of penetration testing methodologies have been around for a while and continue to be updated as new threats emerge. The following are some of the most common:

- **Penetration Testing Execution Standard (PTES):** www.pentest-standard.org/index.php/Main_Page
- **NIST Technical Guide to Information Security Testing and Assessment:** https://csrc.nist.gov/publications/detail/sp/800-115/final
- **OWASP Web Security Testing Guide:** https://owasp.org/www-project-web-security-testing-guide
- **Open Source Security Testing Methodology Manual (OSSTMM):** www.isecom.org/research.html

Figure 8-1 shows the penetration testing lifecycle based on the Penetration Testing Execution Standard guidance.

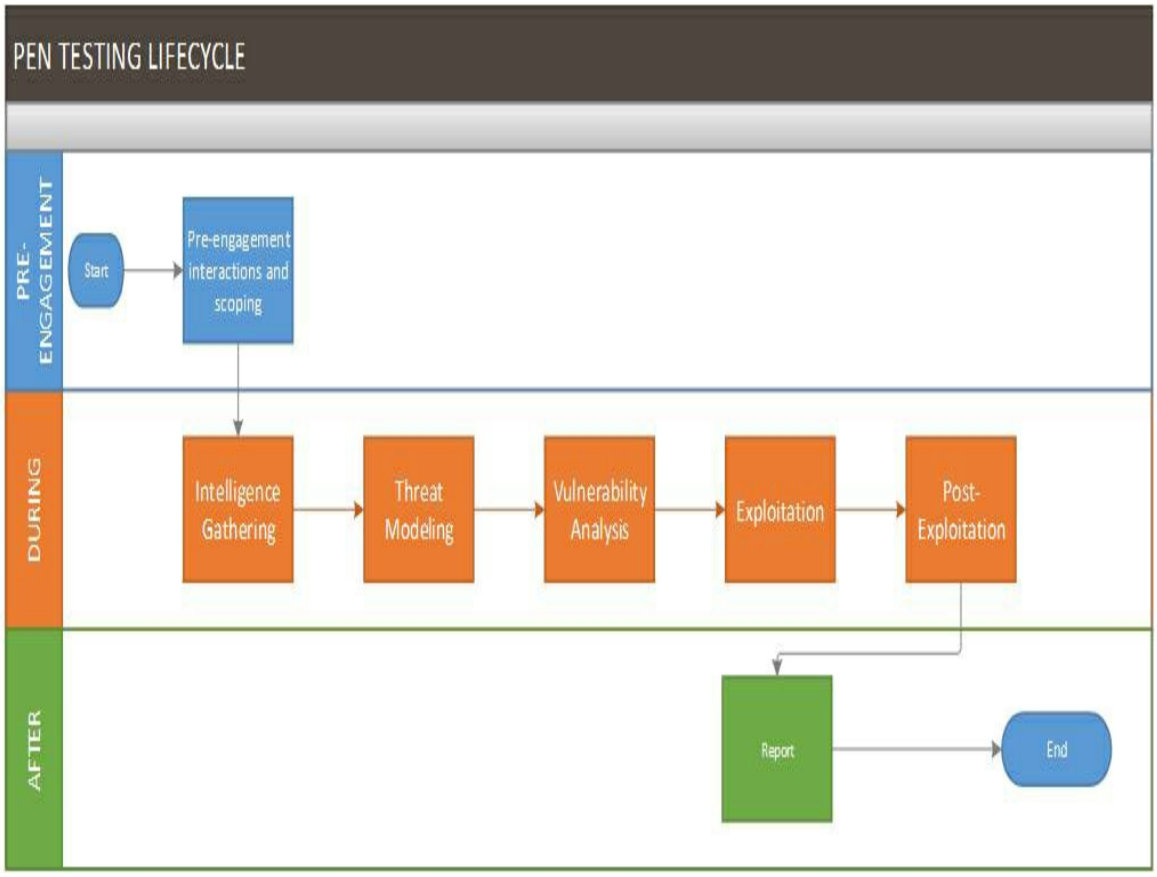


Figure 8-1 The Penetration Testing Lifecycle

The penetration testing lifecycle has three major milestones (as shown in Figure 8-1): the pre-engagement tasks, the actual testing, and the report delivered to the customer after testing. The pre-engagement tasks include items such as contract negotiations, the statement of work (SOW), scoping, and the rules of engagement.

The ***rules of engagement*** documentation specifies the conditions under which the security penetration testing engagement will be conducted. You need to document and agree upon these rules of engagement conditions with the client or an appropriate stakeholder. The following elements are typically included in a rules of engagement document:

- Testing timeline
- Location of the testing
- Time window of the testing
- Preferred method of communication
- The security controls that could potentially detect or prevent testing
- IP addresses or networks from which testing will originate
- The scope of the engagement

During the penetration testing, the ethical hacker performs passive and active reconnaissance to find vulnerabilities. Passive and active reconnaissance are discussed later in this chapter. Based on the findings from the reconnaissance phase, the ethical hacker then starts finding security vulnerabilities and demonstrates how an attacker could exploit such vulnerabilities. After exploiting a vulnerability, the ethical hacker could also demonstrate how an attacker could perform post-exploitation activities such as

- ***Lateral movement* and *Pivoting*:** Lateral movement (also referred to as *pivoting*) is a post-exploitation technique that can be performed using many different methods. The main goal of lateral movement is to move from one device to another to avoid detection, steal sensitive data, and maintain access to these devices to exfiltrate the sensitive data. Lateral movement involves scanning a network for other systems, exploiting vulnerabilities in other systems, compromising credentials, and collecting sensitive information for exfiltration. Lateral movement is possible if an organization does not segment its network in a proper way. Network segmentation is therefore very important. After compromising a system, you can use basic port scans to identify systems or services of interest that you can further attack in an attempt to compromise valuable information.
- ***Privilege escalation*:** This is the process of elevating the level of authority (privileges) of a compromised user or a compromised application. It is done to further perform actions on the affected system

or any other systems in the network. It is possible to perform privilege escalation in a few different ways. An attacker may be able to compromise a system by logging in with a nonprivileged account. Subsequently, the attacker can go from that unprivileged (or less privileged) account to another account that has greater authority. It is also possible to perform privilege escalation by "upgrading," or elevating, the privileges of the same account.

- ***Persistence*:** After the exploitation phase, you need to maintain a foothold in a compromised system to perform additional tasks such as installing and/or modifying services to connect back to the compromised system. You can maintain persistence of a compromised system in a number of ways, including the following:
  - Creating a bind or reverse shell
  - Creating and manipulating scheduled jobs and tasks
  - Creating custom daemons and processes
  - Creating new users
  - Creating additional backdoors

Tip

When you maintain persistence in a compromised system, you can take several actions, such as the following:

- Uploading additional tools
- Using local system tools
- Performing ARP scans and ping sweeps
- Conducting DNS and directory services enumeration
- Launching brute-force attacks
- Performing additional enumeration and system manipulation using management protocols (for example, WinRM, WMI, SMB, SNMP) and compromised credentials
- Executing additional exploits

Ethical hackers or pen testers also need to be able to cover their tracks. Many tools can leave behind residual files or data that you need to be sure to clean from the target systems after the testing phases of a penetration testing engagement are complete. It is also very important to have the client or system owner validate that the ***cleanup*** effort is sufficient. This is not always easy to accomplish, but providing a comprehensive list of activities performed on any systems under test will help.

Following are some items you will want to be sure to clean from systems:

- User accounts created
- Shells spawned on exploited systems
- Database input created by automated tools or manually
- Any tools installed or run from the systems under test

Bug Bounties vs. Penetration Testing

Bug bounties have become very popular in the last several years. In a bug bounty program, an organization provides recognition and compensation to security researchers for reporting security vulnerabilities (which are basically *bugs* in code or hardware). The goal of bug bounty programs is to allow underlying organizations to crowdsource the way they find vulnerabilities in their systems and infrastructure in a scalable way.

These programs allow an underlying organization to learn about bugs and fix them before an attacker can abuse them. Several bug bounty platforms (or brokers) help crowdsource and manage a bug bounty program. Examples of bug bounty platforms/companies include

- HackerOne (www.hackerone.com)
- BugCrowd (www.bugcrowd.com)

- Intigriti (www.intigriti.com)

Security researchers and ethical hackers often participate either full-time or part-time in these bug bounty programs.

Tip

For a list of references and resources about bug bounty Programs, platforms, tools, and tips, see https://github.com/The-Art-of-Hacking/h4cker/tree/master/bug-bounties.

Passive and Active Reconnaissance

Reconnaissance can be passive or active. ***Passive reconnaissance*** can be carried out by an attacker just researching information about the victim's public records, social media sites, and other technical information, such as DNS, whois, and sites such as Shodan (www.shodan.io). Searching through these public records is often referred to as ***open-source intelligence (OSINT)*** gathering. The ethical hacker or pen tester can use tools such as Maltego, Recon-ng, theHarvester, SpiderFoot, OWASP Amass, and many others to accelerate this research.

Tip

The download links of the aforementioned OSINT tools (along with several others) are included in the GitHub repository at https://github.com/The-Art-of-Hacking/h4cker/tree/master/osint. This GitHub repository includes a curated list of numerous references and resources about OSINT and passive reconnaissance tools and methodologies.

For instance, the Shodan search engine is a powerful database of prescanned networked devices connected to the Internet. It consists of scan results including banners collected from port scans of public IP addresses, with fingerprints of services like Telnet, FTP, HTTP, and other applications.

Shodan creates a risk profile by providing both attackers and defenders with a prescanned inventory of devices connected to public IP addresses on the Internet. For example, when a new vulnerability is discovered and published, an attacker can quickly and easily search Shodan for vulnerable devices and then launch an attack. Attackers can also search the Shodan database for devices with poor configurations or other weaknesses, all without actively scanning. Using Shodan search filters, a user can really narrow down search results, by country code or classless interdomain routing (CIDR) netblock, for example. Shodan application programming interfaces (APIs) and some basic scripting can enable many search queries and subsequent actions (for example, a weekly query of newly discovered IPs scanned by Shodan on a CIDR netblock that runs automatically and is emailed to the security team). Remember that public IP addresses are constantly probed and scanned already. By using Shodan, you are not scanning because Shodan has already scanned these IPs. Shodan is a tool, and it can be used for good or evil. To mitigate risk, you can take tangible steps like registering for a free Shodan account, searching for your organization's public IPs, and informing the right network and security people of the risks of your organization's Shodan exposure. You can learn more at www.shodan.io.

Active reconnaissance is carried out mostly by using network and vulnerability scanners. The following are commercial and open-source application, port, and vulnerability scanners:

- AppScan by IBM
- Burp Suite Professional by PortSwigger
- Nessus by Tenable Network Security
- Netsparker by Mavituna Security
- Nexpose by Rapid7
- Nmap (open-source port scanner)
- Nikto (open-source web application scanner)
- OWASP Zed Attack Proxy (open-source web application scanner, proxy, and attack platform maintained by the Open Web Application Security Project [OWASP])
- Qualys

- Retina Web Security Scanner by eEye Digital Security
- Sentinel by WhiteHat
- Veracode Web Application Security by Veracode
- VUPEN Web Application Security Scanner by VUPEN Security
- WebApp360 by nCircle

One of the main purposes of reconnaissance is to footprint (perform ***footprinting*** on) an application, system, or network to find vulnerabilities that could potentially be exploited. Attackers and pen testers have used several creative methods to perform reconnaissance. For example, even ***drones*** have been used to eavesdrop and monitor wireless networks. The term ***war driving*** refers to the ability of an attacker to just drive around and get a huge amount of information over a very short period of time. The attacker could obtain observations about wireless access points, underlying service set identifiers (SSIDs), and even Bluetooth and cellular communications. A good example of a repository and application that has been used to store and analyze war driving records is Wigle (https://wigle.net/). Any individual can download the Wigle mobile app and automatically upload information about Wi-Fi networks, cell towers, and Bluetooth devices.

A similar concept is ***war flying***. In war flying an attacker can fly drones or similar devices to obtain information about a wireless network or even collect pictures of facilities in some cases.

Tip

Another great resource to learn about real-life adversarial techniques used for passive and active reconnaissance (as well as the complete lifecycle of a cybersecurity attack) is the MITRE ATT&CK framework, which can be accessed at https://attack.mitre.org.

Exercise Types

Several terms have been used to define the different penetration testing or cybersecurity exercise types. The following terms are the most common:

- ***Red team*:** Individuals who are performing adversarial simulation and penetration testing. However, a "true red team engagement" goes beyond the traditional scope of a penetrating testing. For example, red teamers can also demonstrate how an attacker could infiltrate a building and perform advanced social engineering attacks. Red teamers also create exploits and their own tools.
- ***Blue team*:** Defenders of the organizations. Blue teams typically include the computer security incident response team (CSIRT) and information security (InfoSec) teams. The expression "blue teams versus red teams" is adopted from the military, where the red team is typically the "offensive" team and blue is the "defensive" team.
- ***Purple team*:** Individuals who perform different tactics to maximize the effectiveness of red and blue team operations. If you combine the colors red and blue, you end up with purple. Similarly in security, purple teams integrate the defensive capabilities of a blue team with the adversarial techniques used by the red team. In most cases, the purple team is not a separate team, but a solid dynamic between the blue and read teams.
- ***White team*:** A team that blends all previous colors together. White teams are individuals who are focused on governance, management, risk assessment, and compliance.

Tip

In teaming security assessments, the blue team members are the defenders. It is their job to counter the red team and keep them from accomplishing their mission. This team has the additional advantage of measuring and improving alerting and response. The red team members act as the adversaries. They are the ones

attacking and trying to remain unnoticed. The white team members are neutral. They are the referees who define the goals and rules and also adjudicate the exercise. The goals of the purple team are to effectively combine the skills and knowledge of both the red and blue teams to achieve maximum effectiveness.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 8-2 lists a reference of these key topics and the page number on which each is found.

Table 8-2 Key Topics for Chapter 8

Key Topic Element	Description	Page Number
List	Understanding known environment, unknown environment, or partially known environment penetration testing	
Paragraph	Defining rules of engagement	
List	Defining lateral movement, pivoting, privilege escalation, and persistence	
Paragraph	Understanding how attackers and penetration testing cover their tracks and perform cleanup	
Paragraph	Surveying what bug bounties are	
Paragraph	Defining footprinting, war driving, and war flying	
List	Defining red, blue, purple, and white teams	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

penetration testing

known environment

unknown environment

partially known environment

rules of engagement

lateral movement

pivoting

privilege escalation

persistence

cleanup

bug bounties

passive reconnaissance

open-source intelligence (OSINT)

active reconnaissance

footprinting

drones

war driving

war flying

red team

blue team

purple team

white team

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. You were hired to perform a penetration test against a set of applications. After the exploitation phase, you need to maintain a foothold in a compromised system to perform additional tasks such as installing and/or modifying services to connect back to the compromised system. This process is referred to as _____.
2. What is the process of elevating the level of authority (privileges) of a compromised user or a compromised application?
3. What is the term used to define the type of testing where the penetration testers may be provided credentials but not full documentation of the network infrastructure?
4. What is the term used when an organization provides recognition or compensation to security researchers and ethical hackers who report security vulnerabilities or bugs? Often organizations can use brokers and companies that manage the compensation and communication with the security researchers.
5. OSINT is used in the _____ reconnaissance phase of the penetration testing lifecycle.

Part II: Architecture and Design

Chapter 9. Understanding the Importance of Security Concepts in an Enterprise Environment

This chapter covers the following topics related to Objective 2.1 (Explain the importance of security concepts in an enterprise environment) of the CompTIA Security+ SY0-601 certification exam:

- Configuration management
  - Diagrams
  - Baseline configuration
  - Standard naming conventions
  - Internet protocol (IP) schema
- Data sovereignty
- Data protection
  - Data loss prevention (DLP)
  - Masking
  - Encryption at rest, in transit/motion, and in processing
  - Tokenization
  - Rights management
- Geographical considerations
- Response and recovery controls
- Secure Sockets Layer (SSL)/Transport Layer Security (TLS) inspection
- Hashing
- API considerations

- Site resiliency
    - Hot site
    - Cold site
    - Warm site
- Deception and disruption
    - Honeypots
    - Honeyfiles
    - Honeynets
    - Fake telemetry
    - DNS sinkhole

This chapter starts with an overview of best practices for system configuration management and then details different approaches for data sovereignty and protection. You learn about different geographical considerations for data protection, as well as response and recovery controls. This chapter also covers the principles of Secure Sockets Layer (SSL)/Transport Layer Security (TLS) inspection, hashing, and considerations to protect APIs. You also learn about the different methods for site resiliency and techniques for deception and disruptions (including honeypots, honeyfiles, honeynets, fake telemetry, DNS sinkholes, and others).

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Chapter Review Activities" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 9-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 9-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Configuration Management	1–3
Data Sovereignty and Data Protection	4–7
Site Resiliency	8–9
Deception and Disruption	10–12

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which of the following is a primary goal of configuration management?

a. Maintaining computer systems, servers, network infrastructure, and software in a desired, consistent state

b. Reducing the cost of acquiring computer systems, servers, network infrastructure, and software used for information security

c. Ensuring that any changes done to the infrastructure do not affect the underlying organization's IT budget

d. All of these answers are correct.

2. After a minimum desired state of security is defined, _____ should be taken to assess the current security state of computers, servers, network devices, and the network in general.

a. Network diagrams

b. IPv4 schemas

c. Baselines

d. None of these answers are correct.

3. Which of the following is a benefit of standard naming conventions in an IT infrastructure?

a. Appropriate naming conventions are used to avoid conflicts and to be able to correlate data among disparate systems.

b. Appropriate naming conventions are used to reduce unnecessary spending of IT infrastructure.

c. Appropriate naming conventions are used to better create IPv6 network schemas and for data sovereignty.

d. None of these answers are correct.

4. Which of the following are privacy laws or regulations? (Choose two).

a. PCI-DSS

b. CCPA

c. GDPR

d. FedRamp

5. Which of the following is a type of software or hardware-based data loss prevention solution?

a. Endpoint DLP systems

b. Network DLP systems

c. Storage DLP systems

d. All of these answers are correct.

6. You were hired to deploy a system to prevent unauthorized use and transmission of confidential information. What should you prioritize to protect and encrypt?

a. Data at rest

b. Data in use

c. Data in motion

d. All of the answers are correct.

7. Which of the following are used in digital signatures, in file and message authentication, and as a way to protect and verify the integrity of sensitive data?

a. Data masking

b. Tokenization

c. Hashes

d. Redaction

8. Which term is used when you have a near duplicate of the original site of the organization that can be up and running within minutes?

a. Hot site

b. Warm site

c. Cluster site

d. Cold site

9. What do you call a redundant site that has tables, chairs, bathrooms, and possibly some technical setup, but a lot of configuration of computers and data restoration is necessary before the site can be properly utilized?

a. Hot site

b. Warm site

c. Cluster site

d. Cold site

10. Which term is used to categorize a group of computers used to attract and trap potential adversaries to counteract and analyze an attack?

a. Honeypot

b. Honeynet

c. Honeyfile

d. None of these answers are correct.

11. A security analyst creates a file called passwords.txt to lure attackers to

access it. Which term is used for this technique?

a. Honeynet

b. Honeypot

c. Honeyfarm

d. Honeyfile

12. In a _____ you configure one or more DNS servers to provide false results to attackers and redirect them to areas in the network where you can observe their tactics and techniques.

a. DNS sinkhole

b. DNS tunnel

c. DNS Zone transfer

d. None of these answers are correct.

Foundation Topics

Configuration Management

Configuration management is an ongoing process created with the goal of maintaining computer systems, servers, network infrastructure, and software in a desired, consistent state. One of the primary goals of configuration management is to ensure that your infrastructure performs as it's expected to as changes are made over time.

Several key elements are used in the process of configuration management:

- ***Diagrams*** **and other documentation:** A good configuration management process helps to avoid small or large changes going undocumented. These undocumented changes can lead to poor performance, inconsistencies, or noncompliance and negatively impact business operations and security. When poorly documented changes are made within your infrastructure, they add to instability and downtime.

Having good network diagrams and well-written and up-to-date documentation is crucial and allows you to not only troubleshoot problems but also respond quickly to security incidents.

- ***Baseline configuration*:** After a minimum desired state of security is defined, baselines should be taken to assess the current security state of computers, servers, network devices, and the network in general. Baseline configurations should be properly documented and reviewed to include a set of specifications for information systems or configuration items within those systems. Baseline configurations are used by security professionals, along with network and system administrators, as a basis for future deployments, releases, or changes to information systems and applications. A baseline configuration could include information about standard software packages installed on endpoint systems, servers, network infrastructure devices, mobile devices, or applications and infrastructure hosted in the cloud. These baseline configurations should also include current version numbers and patch information on operating systems and applications, and configuration parameters, network topology, and the logical placement of those components within the system architecture. You should always review configuration baselines to make sure that they are still relevant. New baselines should be created as organizational information systems change over time.
- ***Standard naming conventions*:** You should make sure that your organization has appropriate naming conventions for describing IT infrastructure, applications, and users. Appropriate naming conventions are used to avoid conflicts and to be able to correlate data among disparate systems.
- ***Internet protocol (IP) schema*:** Similar to standard naming conventions, having a proper IPv4 or IPv6 schema will help avoid conflicts within your on-premises network or cloud deployments and to be able to correlate data among disparate systems. For example, when using RFC 1918 private IP addresses, you should perform proper planning so that the IP network scheme can be more organized, easier to set up, and easier to troubleshoot than user and network services. Identify what subnets are used for wired and wireless users, as well as virtual private network (VPN) users. Wireless access may require additional subnets for guest access, quarantine of nonsecure devices, IoT devices, and so

on. In addition, you may have IP address subnets dedicated for voice over IP (VoIP) devices, separate for printers and IoT devices. Configuration management systems let you consistently define system settings, as well as build and maintain those systems according to those baseline settings.

Data Sovereignty and Data Protection

Data sovereignty refers to any information (data) that has been converted and stored in a digital form. Many laws and regulations have regulated how organizations should handle their customer and employee data. One of the main concerns around data sovereignty is privacy. For example, the General Data Protection Regulation (GDPR) is a regulation in the European Union (EU) and the European Economic Area (EEA) focused on data protection and privacy. Another example is the California Consumer Privacy Act (CCPA). These regulations give consumers the right to know what personal information is being collected by companies, government, and any other organizations. Consumers can also access the personal information that is collected and request that it be deleted, as well as to know whether their personal information is being shared (and if so, with whom). In addition, consumers can opt out of the sale of their personal information.

Another concept that is crucial for ***data protection*** is ***data loss prevention (DLP)*** systems. DLP is a concept that refers to the monitoring of data in processing, data in transit/motion, and data at rest. A DLP system performs content inspection and is designed to prevent unauthorized use of data as well as prevent the leakage of data outside the computer (or network) where it resides. DLP systems can be software- or hardware-based solutions and come in three varieties:

- **Endpoint DLP systems:** These systems run on an individual computer and are usually software-based. They monitor data in processing, such as email communications, and can control what information flows

between various users. These systems can also be used to inspect the content of USB-based mass-storage devices or block those devices from being accessed altogether by creating rules within the software.

- **Network DLP systems:** These software- or hardware-based solutions are often installed on the perimeter of the network. They inspect data that is in transit/motion.
- **Storage DLP systems:** These systems are typically installed in data centers or server rooms as software that inspects data at rest.

Most cloud providers offer cloud-based DLP solutions to protect against data breaches and misuse of data. These solutions often integrate with software, infrastructure, and platform services and can include any of the systems mentioned previously. Cloud-based DLP is necessary for companies that have increased bring-your-own-device (BYOD) usage, and that store data and operate infrastructure within the cloud.

As with host intrusion detection systems (HIDS) and network intrusion detection systems (NIDS), DLP solutions must be accurate and updated to reduce the number of false positives and false negatives. Most systems alert the security administrator if there is a possibility of data leakage. However, it is up to the administrator to determine whether the threat is real.

Secure Sockets Layer (SSL)/Transport Layer Security (TLS) Inspection

Transport Layer Security Inspection (TLSI) is a security process that allows organizations to decrypt traffic, inspect the decrypted content for threats, and then reencrypt the traffic before it enters or leaves the network. TLSI is also known as "TLS break and inspect." TLS replaces the Secure Sockets Layer (SSL) protocol. In the past, TLSI was referred to as ***SSL Inspection (SSLI)***. Because newer implementations use TLS instead of SSL, this chapter uses the term *TLSI* to refer to this type of inspection. Using TLSI enhances visibility within some security products (such as IDS and DLP systems at the network edge) but also introduces new threats.

One of the main threats introduced when using TLSI is the potential abuse of the certificate authority (CA). Attackers can use this CA to issue unauthorized certificates trusted by the TLS clients. Abuse of a trusted CA could allow an attacker to sign malicious code to security controls and monitoring capabilities or to deploy malicious services that impersonate legitimate services to endpoints.

Some organizations can configure a policy to enforce traffic to be decrypted and inspected only "as authorized" to ensure that decrypted traffic is contained in an out-of-band and isolated segment of the network. This technique can be used to prevent unauthorized access to the decrypted traffic.

Tip

As a mitigation, you should break and inspect TLS traffic only once within the organization's network. Redundant TLSI (where traffic is decrypted and inspected more than once) should not be performed. Inspecting multiple times can greatly complicate the ability to diagnose network issues with TLS traffic.

API Considerations

Application programming interfaces (APIs) are used in most modern applications. Organizations should evaluate different ***API considerations*** to make sure that APIs and underlying implementations are secure. If not configured or managed correctly, they can expose sensitive data to attackers. Furthermore, in the past there were several major incidents where the privacy of individuals was compromised. For example, several years ago Cambridge Analytica used Facebook's APIs to obtain information about millions of Facebook users. The misuse of similar APIs has sparked many privacy concerns from consumers and lawmakers. Nowadays, APIs are used in mobile apps, games, social networking platforms, dating apps, news websites, e-commerce sites, video and music streaming services, mobile payment systems, and many other implementations. When a third-party organization (such as an app developer or advertiser) obtains access to data through APIs, it may also gain access to very sensitive personal information. Some of these past events also sparked the creation of enhanced privacy laws

in the United States, Europe, and worldwide.

Data Masking and Obfuscation

Data masking (otherwise known as data obfuscation) is the act of hiding sensitive information/data with specific characters or other data in order to protect it. For example, data masking (and obfuscation) has been used to protect personally identifiable information (PII) or commercially sensitive data from unauthorized users and attackers.

The following techniques have been used for data masking and obfuscation:

- **Substitution:** In this technique you substitute the original data with another authentic-looking value. For instance, you may replace a Social Security number or credit card number with a fake number. Figure 9-1 illustrates a basic example of substitution.

Original Credit Card Number

Substitute number that looks real

Figure 9-1 Data Masking Using Substitution

Figure 9-2 shows another example of masking by revealing only the last four digits of the credit card.

Original Credit Card Number

Masking most of the data

Figure 9-2 Additional Data Masking Example

- ***Tokenization*:** This technique is used mostly when protecting data at rest. It is the process of randomly generating a token value for plaintext data and store the mapping in a database. Tokenization is difficult to scale securely due to the performance and size of the underlying database. Tokenized data is difficult to exchange, since it requires direct access to the database (or token vault). Encryption provides a better way to scale and exchange sensitive data in a secure manner.

Encryption at Rest, in Transit/Motion, and in Processing

What do you need to encrypt? Without a doubt, you need to encrypt data, but more specifically three types of data: data in use, data at rest, and data in transit.

Figure 9-3 illustrates the concepts of encrypting data at rest, in transit/motion, and in use/processing.

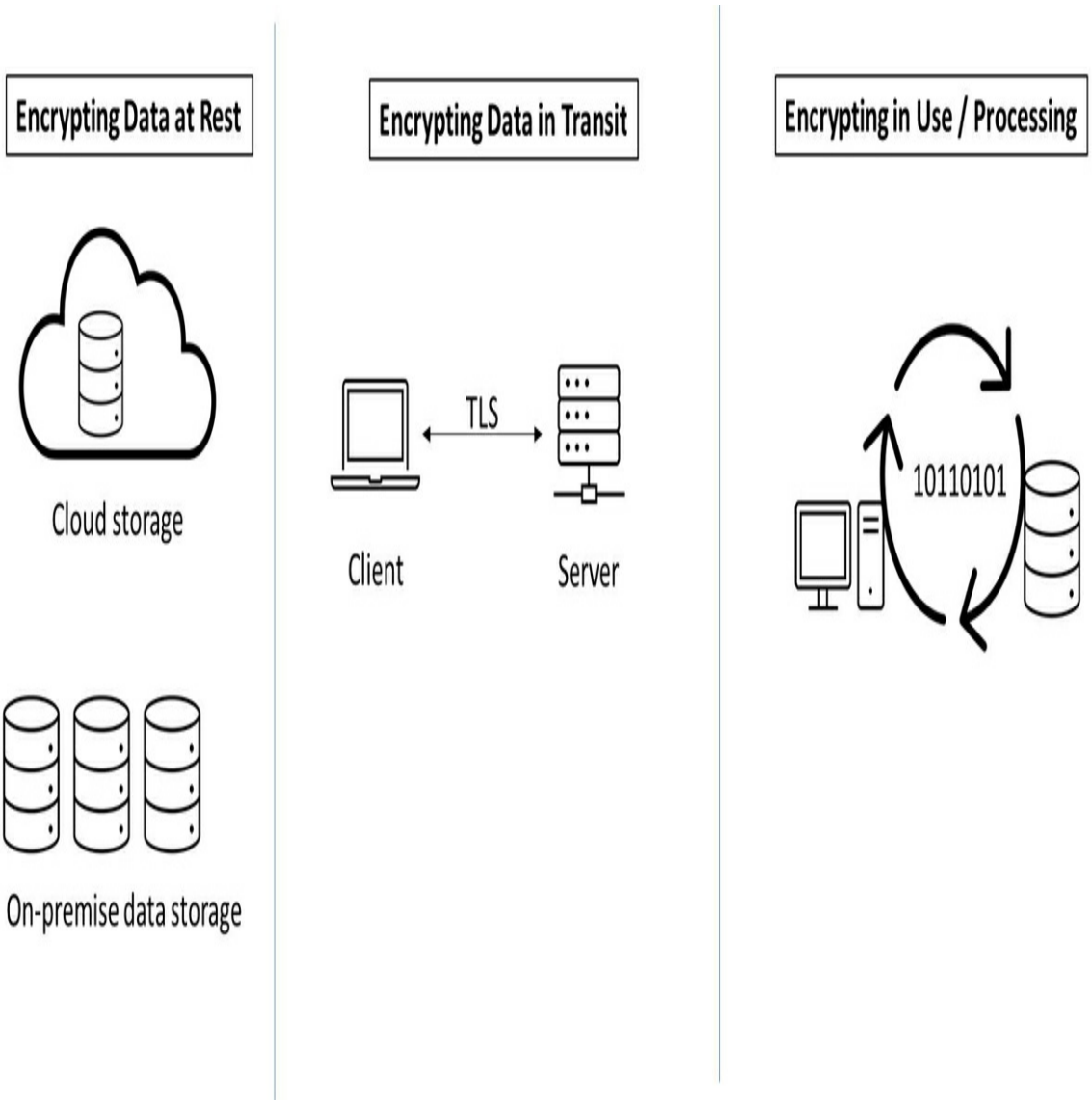


Figure 9-3 Encrypting Data at Rest, in Transit/Motion, and in Use/Processing

Key Topic

Data in use/processing can be described as actively used data undergoing constant change; for example, it could be stored in databases or spreadsheets. ***Data at rest*** is inactive data that is archived—backed up or stored in cloud storage services. ***Data in transit*** (also known as data in motion) is data that crosses the network or data that currently resides in computer memory.

Hashing

A ***hash*** is a summary of a file or message, often in numeric format. Hashes are used in digital signatures, in file and message authentication, and as a way to protect the integrity of sensitive data—for example, data entered into databases or perhaps entire hard drives. A hash is generated through the use of a hash function to verify the integrity of the file or message, most commonly after transit over a network. A *hash function* is a mathematical procedure that converts a variable-sized amount of data into a smaller block of data. The hash function is designed to take an arbitrary data block from the file or message, use that as an input, and from that block produce a fixed-length hash value. Basically, the hash is created at the source and is recalculated and compared with the original hash at the destination.

Because the hash is a condensed version of the file/message, or a portion of it, it is also known as a message digest. It provides integrity to data so that a user knows that the message is intact, hasn't been modified during transit, and comes from the source the user expects. A hash can fall into the category of a *one-way function*. This means it is easy to compute when generated but difficult (or impossible) to compute in reverse. In the case of a hash, a condensed version of the message, initial computation is relatively easy (compared to other algorithms), but the original message should not be re-created from the hash. Contrast this concept to encryption methods that indeed can be reversed. A hash can be created without the use of an algorithm, but generally, the ones used in the field require some kind of cryptographic algorithm.

Note

In Chapter 16, "Summarizing the Basics of Cryptographic Concepts," you will learn the details about different hashing algorithms such as MD5 and different versions of SHA. You will also learn which flavors of those algorithms are recommended and which should be avoided (such as MD5).

Rights Management

Digital rights management (DRM) is the name given to a set of access control technologies that are used to control the use of proprietary hardware, software, and copyrighted works. DRM solutions are used to restrict the use, modification, and distribution of copyrighted works and the underlying systems used to enforce such policies.

Traditional DRM policies include restrictive licensing agreements and solutions that encrypt expressive material or embedding a digital tag designed to control access and reproduction of information. Software companies and content publishers typically enforce their own access policies on content (that is, restrictions on copying or viewing such content or software). Hardware manufacturers have also expanded the usage of DRM to more traditional hardware products.

Digital rights management policies and processes are not universally accepted. Some argue that there is no substantial evidence that DRM helps to completely prevent copyright infringement. The following section covers some of the geographical considerations for data protection and DRM.

Geographical Considerations

Many ***geographical considerations*** (depending on where you reside) affect the laws and regulations that have been created to address data privacy and DRM:

- **Data privacy:** Earlier in this chapter, you learned about GDPR (a regulation in the European Union [EU] and the European Economic Area [EEA] focused on data protection and privacy). You also learned about the California Consumer Privacy Act (CCPA). Many other laws and regulations in other countries give consumers the right to know what personal information is being collected by companies, government, and any other organizations. Additional examples include the Australia Privacy Principles (APP), Canada's Personal Information Protection and Electronic Data Act (PIPEDA), and Japan's Personal Information

Protection Act (JPIPA).

- **DRM:** Many laws around the world have been created to criminalize the circumvention of DRM, communication about such circumvention, and the creation and distribution of tools used. Examples of these DRM-related laws are the United States Digital Millennium Copyright Act (US-DMCA) and the European Union's Information Society Directive.

Tip

Different countries might have laws that dictate how personal data is stored and transferred between different geographical locations and countries. In addition, some regulations (such as GDPR) define policies for "third countries" and what personal data can or cannot be transferred between countries. Additional information about GDPR "third countries" definitions and information can be obtained from https://gdpr-info.eu/issues/third-countries/.

Data Breach Response and Recovery Controls

You must have the proper ***response and recovery controls*** in place in the unfortunate event of a data breach. This data breach response and recovery control plan must include the assembly of a team of experts within your organization, as well as legal counsel. You also have to identify a data forensics team (in most cases hired from third-party providers). The forensics team must help determine the source and scope of the breach, as well as collect and analyze evidence to outline remediation steps.

In the case of the data breach, you should notify law enforcement (when applicable) and also notify the affected businesses and individuals based on local and federal laws.

Tip

The United States Federal Trade Commission (FTC) provides detailed guidance and recommendations for post-breach response

and recovery at www.ftc.gov/tips-advice/business-center/guidance/data-breach-response-guide-business.

Site Resiliency

Within the confidentiality, integrity, availability (CIA) triad, redundant sites fall into the category of *availability*. In the case of a disaster, a redundant site can act as a safe haven for your data and users. Redundant sites are sort of a gray area between redundancy and a disaster recovery method. If you have one and need to use it, a "disaster" has probably occurred. But the better the redundant site, the less time the organization loses, and the less it seems like a disaster and more like a failure that you have prepared for. Of course, this outcome all depends on the type of redundant site your organization decides on.

Regarding the types of redundant sites, I like to refer to the story of Goldilocks and the three bears' three bowls of porridge. One was too hot, one too cold, and one just right. Most organizations opt for the warm redundant site as opposed to the hot or cold. Let's look at these three now.

- ***Hot site*:** This site is a near duplicate of the original site of the organization that can be up and running within minutes (maybe longer). Computers and phones are installed and ready to go, a simulated version of the server room stands ready, and the vast majority of the data is replicated to the site on a regular basis in the event that the original site is not accessible to users for whatever reason. Hot sites are used by companies that would face financial ruin in the case that a disaster makes their main site inaccessible for a few days or even a few hours. This is the only type of redundant site that can facilitate a full recovery.
- ***Warm site*:** This site has computers, phones, and servers, but they might require some configuration before users can start working on them. The warm site will have backups of data that might need to be restored; they will probably be several days old. This type of site is chosen the most often by organizations because it has a good amount of configuration yet

remains less expensive than a hot site.

- ***Cold site*:** This site has tables, chairs, bathrooms, and possibly some technical setup—for example, basic phone, data, and electric lines. Otherwise, a lot of configuration of computers and data restoration is necessary before the site can be properly utilized. This type of site is used only if a company can handle the stress of being nonproductive for a week or more.

Although they are redundant, these types of sites are generally known as backup sites because if they are required, a disaster has probably occurred. A good network security administrator tries to plan for, and rely on, redundancy and fault tolerance as much as possible before having to resort to disaster recovery methods.

Deception and Disruption

Honeypots and honeynets attract and trap potential attackers to counteract any attempts at unauthorized access of the network. These solutions isolate the potential attacker in a monitored area and contain dummy resources that look to be of value to the perpetrator. While an attacker is trapped in one of these, the attacker's methods can be studied and analyzed, and the results of those analyses can be applied to the general security of the functional network.

A ***honeypot*** is generally a single computer but could also be a file, group of files, or an area of unused IP address space, whereas a ***honeynet*** is one or more computers, servers, or an area of a network; a honeynet is used when a single honeypot is not sufficient. Either way, the individual computer, or group of servers, will *usually* not house any important company information. Various analysis tools are implemented to study the attacker; these tools, along with a centralized group of honeypots (or a honeynet), are known collectively as a honeyfarm.

One example of a honeypot in action is the spam honeypot. Spam email is one of the worst banes known to network administrators; a spam honeypot

can lure spammers in, enabling network administrators to study the spammers' techniques and habits, thus allowing the network admins to better protect their actual email servers, SMTP relays, SMTP proxies, and so on, over the long term. This solution might ultimately keep the spammers away from the real email addresses because the spammers are occupied elsewhere. Some of the information gained by studying spammers is shared with other network admins or organizations' websites dedicated to reducing spam. A spam honeypot could be as simple as a single email address or as complex as an entire email domain with multiple SMTP servers.

Of course, as with any technology that studies attackers, honeypots also bear risks to the legitimate network. The honeypot or honeynet should be carefully firewalled off from the legitimate network to ensure that the attacker can't break through.

Often, honeypots and honeynets are used as part of a more complex solution known as a network intrusion detection system (NIDS), discussed following a short review of data loss prevention.

Honeyfiles can also be used as bait files intended to lure adversaries to access and then send alarms to security analysts for detection. They can also be used to potentially learn the tactics and techniques used by attackers. For instance, you can create a honeyfile named credentials.txt to lure attackers to access it. Honeyfiles can be used to learn the attackers' tactics, techniques, and behavior without adversely affecting normal operations.

Fake Telemetry

Some organizations have used additional deception techniques, such as ***fake telemetry***, as decoys and breadcrumbs in order to lure and trick attackers. Similarly, attackers have compromised systems that will also generate fake telemetry and reporting data to fool security monitoring systems, analysts in a security operations center (SOC), and evade other security controls that may

be in place.

> **Tip**
>
> The MITRE ATT&CK framework includes examples of these deception and evasion techniques. Examples include the T0856 technique described at https://collaborate.mitre.org/attackics/index.php/Technique/T0856.

DNS Sinkhole

Another deception and disruption technique is the use of ***DNS sinkholes***, or "blackhole DNS servers." In a DNS sinkhole, you configure one or more DNS servers to provide false results to attackers and redirect them to areas in the network where you can observe their tactics and techniques. DNS sinkholes have been used to contain different types of malware such as the infamous WannaCry ransomware and to disrupt certain malicious DNS operations in denial-of-service (DoS) and other attacks. For example, these DNS sinkholes have been used to interrupt DNS resolution to malicious command and control (C2) servers and botnet coordination.

> **Tip**
>
> Adversaries have also used similar techniques to perform DNS poisoning attacks to redirect systems and users to malicious destinations.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 9-2 lists a reference of these key topics and the page number on which each is found.

Table 9-2 Key Topics for Chapter 9

Key Topic Element	Description	Page Number
List	Listing the key elements used in the process of configuration management	
Paragraph	Defining data sovereignty	
Paragraph	Defining data loss prevention (DLP) systems	
Paragraph	Understanding Transport Layer Security Inspection	
Paragraph	Defining data masking and obfuscation	
List	Listing the different techniques used for data masking	
Figure 9-2	Additional data masking example	
Figure 9-3	Encrypting data at rest, in transit/motion, and in use/processing	
Paragraph	Understanding encryption of data in processing, data at rest, and data in transit/motion	

Paragraph	Defining hashing and hashing functions	
Paragraph	Defining digital rights management	
List	Defining hot site, warm site, and cold site in the context of site resiliency	
Paragraph	Defining honeypots and honeynets	
Paragraph	Describing honeyfiles	
Paragraph	Understanding the use of fake telemetry	
Paragraph	Understanding the use of DNS sinkholes	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

configuration management

diagrams

baseline configuration

standard naming conventions

Internet Protocol (IP) schema

data sovereignty

data protection

data loss prevention (DLP)

Transport Layer Security Inspection (TLSI)

SSL Inspection (SSLI)

API considerations

data masking

tokenization

data in use/processing

data at rest

data in transit

hash

digital rights management (DRM)

geographical considerations

response and recovery controls

hot site

warm site

cold site

honeypot

honeynet

honeyfiles

fake telemetry

DNS sinkholes

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. In the context of site resiliency, a _____ will have backups of data that might need to be restored; they will probably be several days old. This type of site is chosen most often by organizations because it has a good amount of configuration yet remains less expensive than a hot site.
2. What can be used as bait files intended to lure adversaries to access and then send alarms to security analysts for detection?
3. What is the name given to a set of access control technologies that are used to control the use of proprietary hardware, software, and copyrighted works?
4. What term is used when data is actively used and undergoing constant change? For instance, data could be stored in databases or spreadsheets and be processed by running applications.
5. What is the process of generating a random value for plaintext data and storing the mapping in a database?

Chapter 10. Summarizing Virtualization and Cloud Computing Concepts

This chapter covers the following topics related to Objective 2.2 (Summarize virtualization and cloud computing concepts) of the CompTIA Security+ SY0-601 certification exam:

- Cloud Models
  - Infrastructure as a service (IaaS)
  - Platform as a service (PaaS)
  - Software as a service (SaaS)
  - Anything as a service (XaaS)
  - Public
  - Community
  - Private
  - Hybrid
- Cloud service providers
- Managed service provider (MSP)/managed security service provider (MSSP)
- On-premises vs. off-premises
- Fog computing
- Edge computing
- Thin client
- Containers
- Microservices/API

- Infrastructure as code
    - Software-defined networking (SDN)
    - Software-defined visibility (SDV)
- Serverless architecture
- Services integration
- Resource policies
- Transit gateway
- Virtualization
    - Virtual machine (VM) sprawl avoidance
    - VM escape protection

These days, more and more organizations are transferring some or all of their server and network resources to the cloud. This effort creates many potential hazards and vulnerabilities that you, as the security administrator, and the cloud provider must address. Top among these concerns are the servers, where all data is stored and accessed. Servers of all types should be hardened and protected from a variety of attacks to keep the integrity of data from being compromised. However, data must also be available. Therefore, you must strike a balance of security and availability. This chapter provides details about several virtualization and cloud computing concepts.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Exam Preparation Tasks" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 10-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 10-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Cloud Models	1–3
Cloud Service Providers	4–5
Cloud Architecture Components	6–8
Virtual Machine (VM) Sprawl Avoidance and VM Escape Protection	9–10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which of the following cloud service models will you use if you want to host applications on virtual machines, deploy load balancers, and use storage buckets?

a. IaaS

b. PaaS

c. SaaS

d. None of these answers are correct.

2. Which of the following cloud deployments is a mix of public and private cloud solutions where multiple organizations can share the public cloud portion?

a. Community cloud

b. PaaS

c. SaaS

d. MSSP cloud

3. Google Drive, Office 365, and Dropbox are examples of which of the following types of cloud service?

a. SaaS

b. IaaS

c. PaaS

d. All of these answers are correct.

4. What type of company or organization provides services to manage your security devices and can also help monitor and respond to security incidents?

a. SaaS provider

b. PaaS provider

c. MSSP

d. Serverless provider

5. Which of the following organizations delivers network, application, system, and management services using a pay-as-you-go model?

a. SaaS provider

b. PaaS provider

c. XaaS provider

d. Managed service provider (MSP)

6. Which term is used to describe an ecosystem of resources and applications in new network services (including 5G and IoT)?

a. SaaS

b. VPC

c. Edge computing

d. None of these answers are correct.

7. Which of the following are computer systems that run from resources

stored on a central server or from the cloud instead of a local (on-premises) system?

a. Thin clients

b. Fog edge devices

c. VPCs

d. Containers

8. Which of the following are technologies and solutions to manage, deploy, and orchestrate containers?

a. Docker Swarm

b. Apache Mesos

c. Kubernetes

d. All of these answers are correct.

9. What condition could occur when an organization can no longer effectively control and manage all the VMs on a network or in the cloud?

a. VM sprawl

b. VM escape

c. Hypervisor escape

d. Hypervisor sprawl

10. Which condition occurs when an attacker or malware compromises one VM and then attacks the hypervisor?

a. VM escape

b. Hypervisor escape

c. VM sprawl

d. Hypervisor sprawl

Foundation Topics

Cloud Models

Cloud computing can be defined as a way of offering on-demand services that extend the capabilities of a person's computer or an organization's network. These might be free services, such as personal browser-based email from various providers, or they could be offered on a pay-per-use basis, such as services that offer data access, data storage, infrastructure, and online gaming. A network connection of some sort is required to make the connection to the cloud and gain access to these services in real time.

Some of the benefits to an organization using cloud-based services include lowered cost, less administration and maintenance, more reliability, increased scalability, and possible increased performance. A basic example of a cloud-based service would be browser-based email. A small business with few employees definitely needs email, but it can't afford the costs of an email server and perhaps does not want to have its own hosted domain and the costs and work that go along with that. By connecting to a free web browser-based service, the small business can obtain near unlimited email, contacts, and calendar solutions. However, there is no administrative control, and there are some security concerns, which are discussed in a little bit.

Cloud computing services are generally broken down into several categories of services:

- ***Software as a service (SaaS)*:** The most commonly used and recognized of the cloud service categories, SaaS offers users access to applications that are provided by a third party over the Internet. The applications need not be installed on the local computer. In many cases these applications are run within a web browser; in other cases the user connects with screen-sharing programs or remote desktop programs. A common example is webmail.

Note

Often compared to SaaS is the application service provider (ASP) model. SaaS typically offers a generalized service to many users. However, an ASP typically delivers a service (perhaps a single application) to a small number of users.

- ***Infrastructure as a service (IaaS)*:** This service offers computer networking, storage, load balancing, routing, and VM hosting. More and more organizations are seeing the benefits of offloading some of their networking infrastructure to the cloud.
- ***Platform as a service (PaaS)*:** This service provides various software solutions to organizations, especially the ability to develop applications in a virtual environment without the cost or administration of a physical platform. PaaS is used for easy-to-configure operating systems and on-demand computing. Often, this utilizes IaaS as well for an underlying infrastructure to the platform. Cloud-based virtual desktop environments (VDEs) and virtual desktop infrastructures (VDIs) are often considered to be part of this service but can be part of IaaS as well.
- ***Anything as a service (XaaS)*:** There are many types of cloud-based services. If they don't fall into the previous categories, they often fall under the category "anything as a service," or XaaS. For instance, a service in which a large service provider integrates its security services into the company's/customer's existing infrastructure is often referred to as security as a service (SECaaS). The concept is that the service provider can provide the security more efficiently and more cost effectively than a company can, especially if it has a limited IT staff or budget. The Cloud Security Alliance (CSA) defines various categories to help businesses implement and understand SECaaS, including encryption, data loss prevention (DLP), continuous monitoring, business continuity and disaster recovery (BCDR), vulnerability scanning, and much more. Periodically, new services will arrive, such as monitoring as a service (MaaS)—a framework that facilitates the deployment of monitoring within the cloud in a continuous fashion.

Tip

NIST's special publication (SP) 800-145 provides a great overview of the definitions of cloud computing concepts. You

can access this document at https://csrc.nist.gov/publications/detail/sp/800-145/final.

Public, Private, Hybrid, and Community Clouds

Organizations use different types of clouds: public, private, hybrid, and community.

- ***Public cloud*:** In this type of cloud, a service provider offers applications and storage space to the general public over the Internet. Examples include free, web-based email services and pay-as-you-go business-class services. The main benefits of this type of service include low (or zero) cost and scalability. Providers of public cloud space include Google, Rackspace, and Amazon.
- ***Private cloud*:** This type of cloud is designed for a particular organization in mind. In it, you, as security administrator, have more control over the data and infrastructure. A limited number of people have access to the cloud, and they are usually located behind a firewall of some sort in order to gain access to the private cloud. Resources might be provided by a third-party or could come from your server room or data center. In other words, the private cloud could be deployed using resources in an on-premises data center or dedicated resources by a third-party cloud provider. An example is Amazon AWS Virtual Private Cloud (https://aws.amazon.com/vpc) and Microsoft Azure private cloud offering (https://azure.microsoft.com/en-us/overview/what-is-a-private-cloud).
- ***Hybrid cloud*:** This type is a mixture of public and private clouds. Dedicated servers located within the organization and cloud servers from a third party are used together to form the collective network. In these hybrid scenarios, confidential data is usually kept in-house.
- ***Community cloud*:** This is another mix of public and private cloud deployments where multiple organizations can share the public portion. Community clouds appeal to organizations that usually have a common

form of computing and storing of data.

The type of cloud an organization uses is dictated by its budget, the level of security it requires, and the amount of manpower (or lack thereof) it has to administer its resources. Although a private cloud can be very appealing, it is often beyond the ability of an organization, forcing that organization to seek a public or community-based cloud. However, it doesn't matter what type of cloud is used. Resources still have to be secured by someone, and you'll have a hand in that security one way or the other.

Cloud Service Providers

A ***cloud service provider (CSP)***, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), might offer one or more of the aforementioned services. CSPs have no choice but to take their security and compliance responsibilities very seriously. For instance, Amazon created a Shared Responsibility Model to define in detail the responsibilities of the AWS customers as well as those of Amazon. You can access the Amazon Shared Responsibility Model at https://aws.amazon.com/compliance/shared-responsibility-model.

Note

The shared responsibility depends on the type of cloud model, whether SaaS, PaaS, or IaaS.

Similar to the CSP is the ***managed service provider (MSP)***, which can deliver network, application, system, and management services using a pay-as-you-go model. An MSP is an organization that can manage your network infrastructure, servers, and in some cases your security devices. Those companies that provide services to manage your security devices and can also help monitor and respond to security incidents are called ***managed security service providers (MSSPs)***.

Tip

Another type of security service provided by vendors is *managed detection and response (MDR)*. MDR evolved from MSSPs to help organizations that lack the resources to establish robust security operations programs detect and respond to threats. MDR is different from traditional MSSP services because it is more focused on threat detection rather than compliance and traditional network security device configuration services. Most of the services offered by MDR providers are delivered using their own set of tools and technologies but are deployed on the customer's premises. The techniques MDR providers use may vary. A few MDR providers rely solely on security logs, whereas others use network security monitoring or endpoint activity to secure the network. MDR solutions rely heavily on security event management; Security Orchestration, Automation and Response (SOAR); and advanced analytics.

Cloud Architecture Components

Cloud computing allows you to store and access your data, create applications, and interact with thousands of other services over the Internet (***off-premises***) instead of your local network and computers (***on-premises***). With a good Internet connection, cloud computing can be executed anytime, anywhere. There are several cloud-related architecture components, described next:

- The concept of fog and edge computing
- Thin clients
- Containers
- Microservices and related APIs

Fog and Edge Computing

The two architectural concepts called fog computing and edge computing are often used interchangeably. However, there are a few differences. The term ***edge computing*** describes an ecosystem of resources and applications in new network services (including 5G and IoT). One of the main benefits is to provide greater network speeds, low latency, and computational power near the user.

Figure 10-1 illustrates the concept of edge computing.

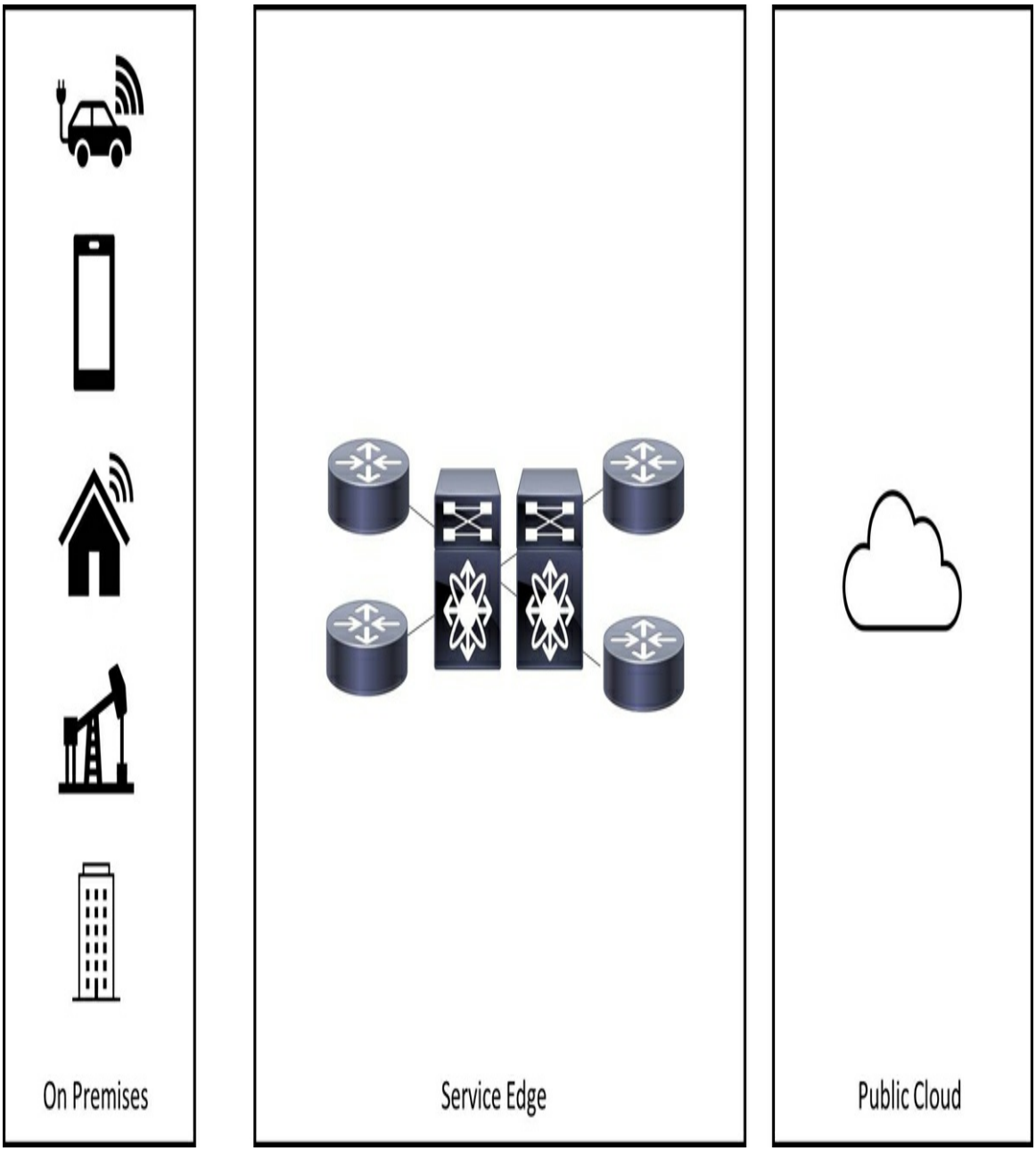


Figure 10-1 Edge Computing

The “edge” in edge computing is the location nearer the consumer (otherwise known as the subscriber) and where data is processed or stored without being backhauled to a central location. This service edge is optimized to provide a low-cost solution, increase performance, and provide a better user experience.

The term ***fog computing*** was originally created by Cisco in the early 2010s. It describes the decentralization of computing infrastructure by "bringing the cloud to the ground" (thus the term *fog*). This architecture enables components of the edge computing concept to easily push compute power away from the public cloud to improve scalability and performance. Fog computing accomplishes this by putting data, compute, storage, and applications near the end user or IoT device.

Other similar and related concepts have been adopted throughout the years. For example, consider the multi-access edge computing (MEC) concept created by the European Telecommunications Standards Institute (ETSI), designed to benefit application developers and content providers. Another term used by some organizations is *cloudlets*, which are micro data centers that can be deployed locally near the data source (in some cases temporarily for emergency response units and other use cases).

Thin Clients

Thin clients are computer systems that run from resources stored on a central server or from the cloud instead of a local (on-premises) system. When you use a thin client, you connect remotely to a server-based computing environment where the applications, sensitive data, and memory are stored.

Thin clients can be used in several ways:

- **Shared terminal services:** Users at thin client stations share a server-based operating system and applications.
- **Desktop virtualization:** A full desktop machine (including the operating system and applications) lives in a virtual machine in an on-premises server or in the cloud (off-premises).
- **Browser-based thin clients:** All applications are accessed within a web browser.

Figure 10-2 shows a user accessing a virtual desktop running Microsoft Windows and custom applications running in the cloud.

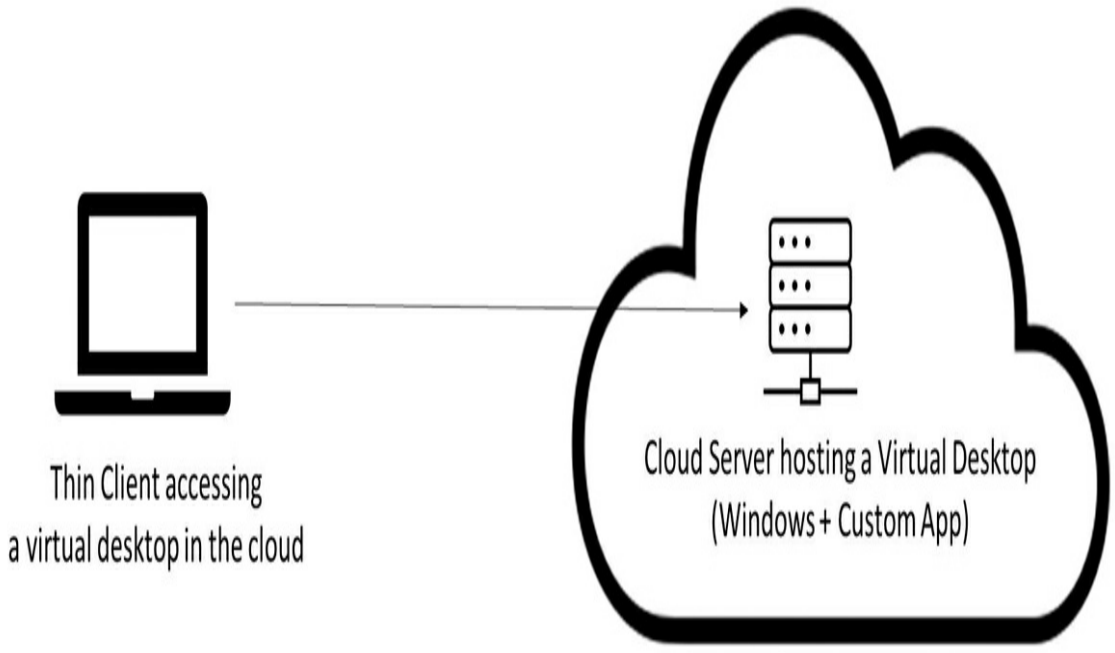


Figure 10-2 Thin Client Accessing a Virtualized Desktop

Containers

Before you can even think of building a distributed system, you must first understand how the container images that contain your applications make up all the underlying pieces of such a distributed system. Applications are normally composed of a language runtime, libraries, and source code. For instance, your application may use third-party or open-source shared libraries such as the Linux Kernel (www.kernel.org), nginx (https://nginx.org), or OpenSSL (www.openssl.org). These shared libraries are typically shipped as shared components in the operating system that you installed on a system. The dependency on these libraries introduces difficulties when an application developed on your desktop, laptop, or any other development machine (dev system) has a dependency on a shared library that isn't available when the program is deployed out to the production system. Even when the dev and production systems share the exact same version of the operating system, bugs can occur when programmers forget to include dependent asset files inside a package that they deploy to production.

The good news is that you can package applications in a way that makes it easy to share them with others. In this case, ***containers*** become very useful. Docker, one of the most popular container runtime engines, makes it easy to package an executable and push it to a remote registry where it can later be pulled by others.

Container registries are available in all of the major public cloud providers (for example, AWS, Google Cloud Platform, and Microsoft Azure) as well as services to build images. You can also run your own registry using open-source or commercial systems. These registries make it easy for developers to manage and deploy private images, while image-builder services provide easy integration with continuous delivery systems.

Container images bundle a program and its dependencies into a single artifact under a root file system. Containers are made up of a series of file system layers. Each layer adds, removes, or modifies files from the preceding layer in the file system. The overlay system is used both when packaging up the image and when the image is actually being used. During runtime, there are various different concrete implementations of such file systems, including aufs, overlay, and overlay2.

Let’s look at an example of how container images work. Figure 10-3 shows three container images: A, B, and C. Container Image B is “forked” from Container Image A. Then, in Container Image B, Python version 3 is added. Furthermore, Container Image C is built on Container Image B, and the programmer adds OpenSSL and nginx to develop a web server and enable TLS.

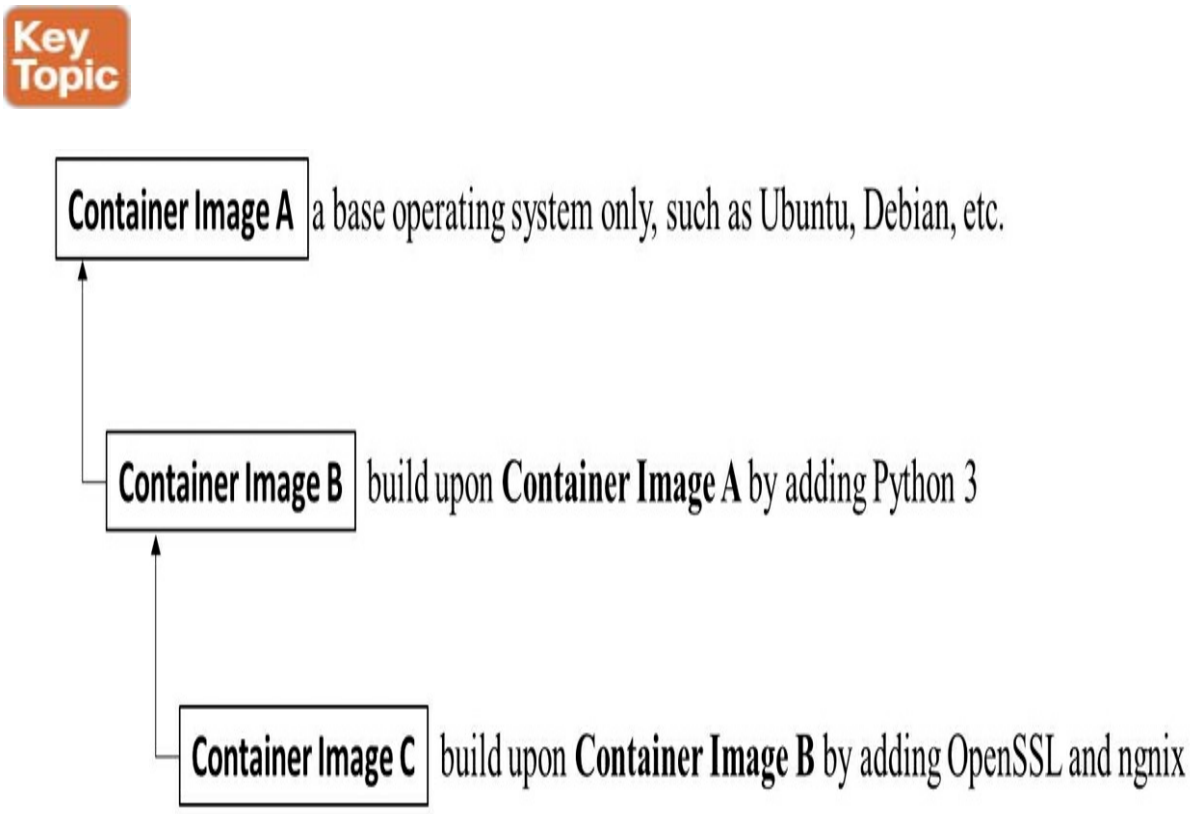


Figure 10-3 How Container Images Work

Abstractly, each container image layer builds on the previous one. Each parent reference is a pointer. The example in Figure 10-3 includes a simple set of containers; in many environments, you will encounter a much larger directed acyclic graph.

Even though the Security+ exam does not cover Docker in detail, it is still good to see a few examples of Docker containers, images, and related commands. Figure 10-4 shows the output of the **docker images** command.

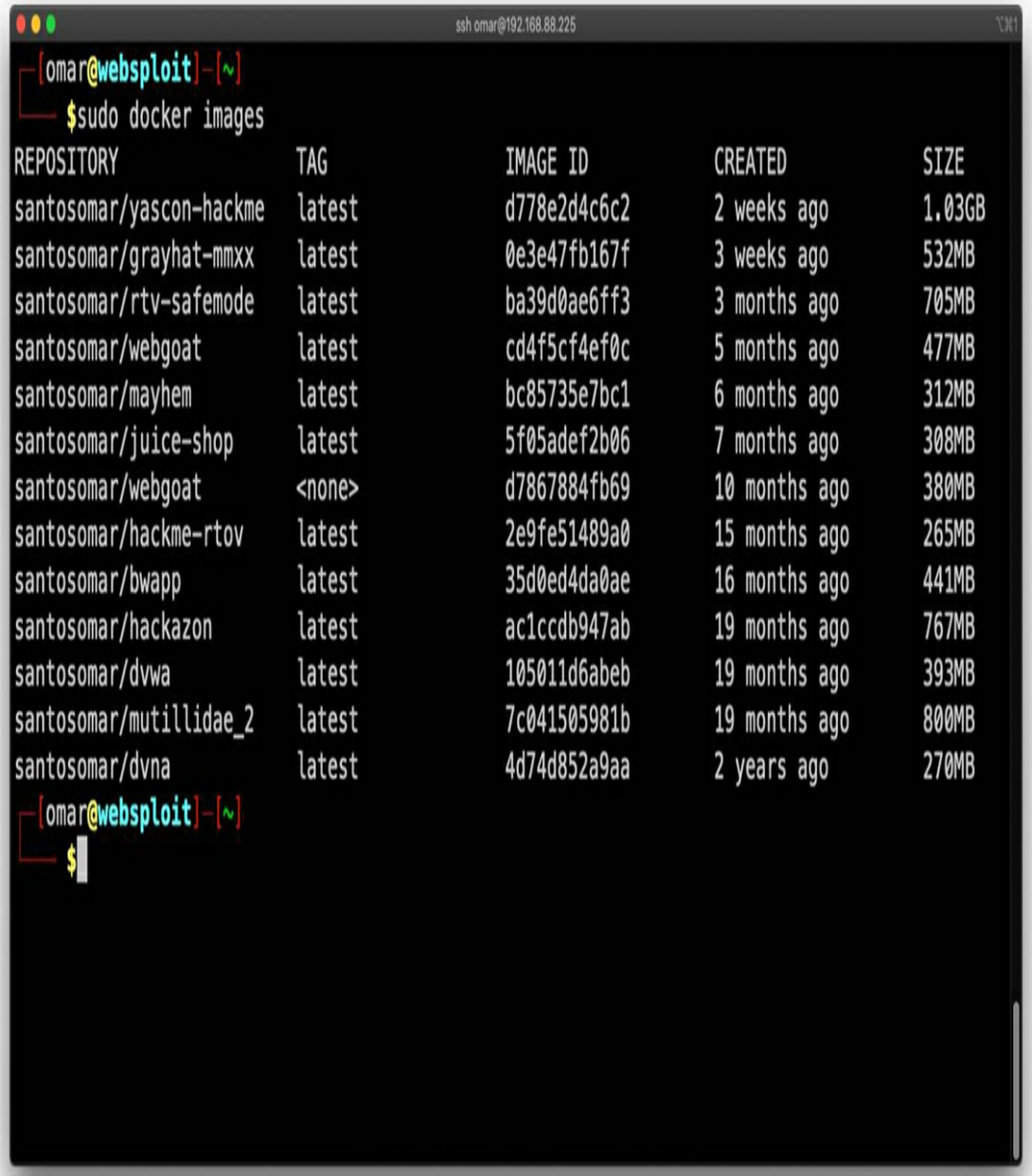


Figure 10-4 Displaying the Docker Images

The Docker images shown in Figure 10-4 are intentionally vulnerable applications that you can also use to practice your skills. These Docker images and containers are included in a VM called WebSploit (websploit.org) by Omar Santos. The VM is built on top of Kali Linux and includes several

additional tools, along with the aforementioned Docker containers. This can be a good tool, not only to get familiarized with Docker, but also to learn and practice offensive and defensive security skills.

Figure 10-5 shows the output of the **docker ps** command used to see all the running Docker containers in a system.

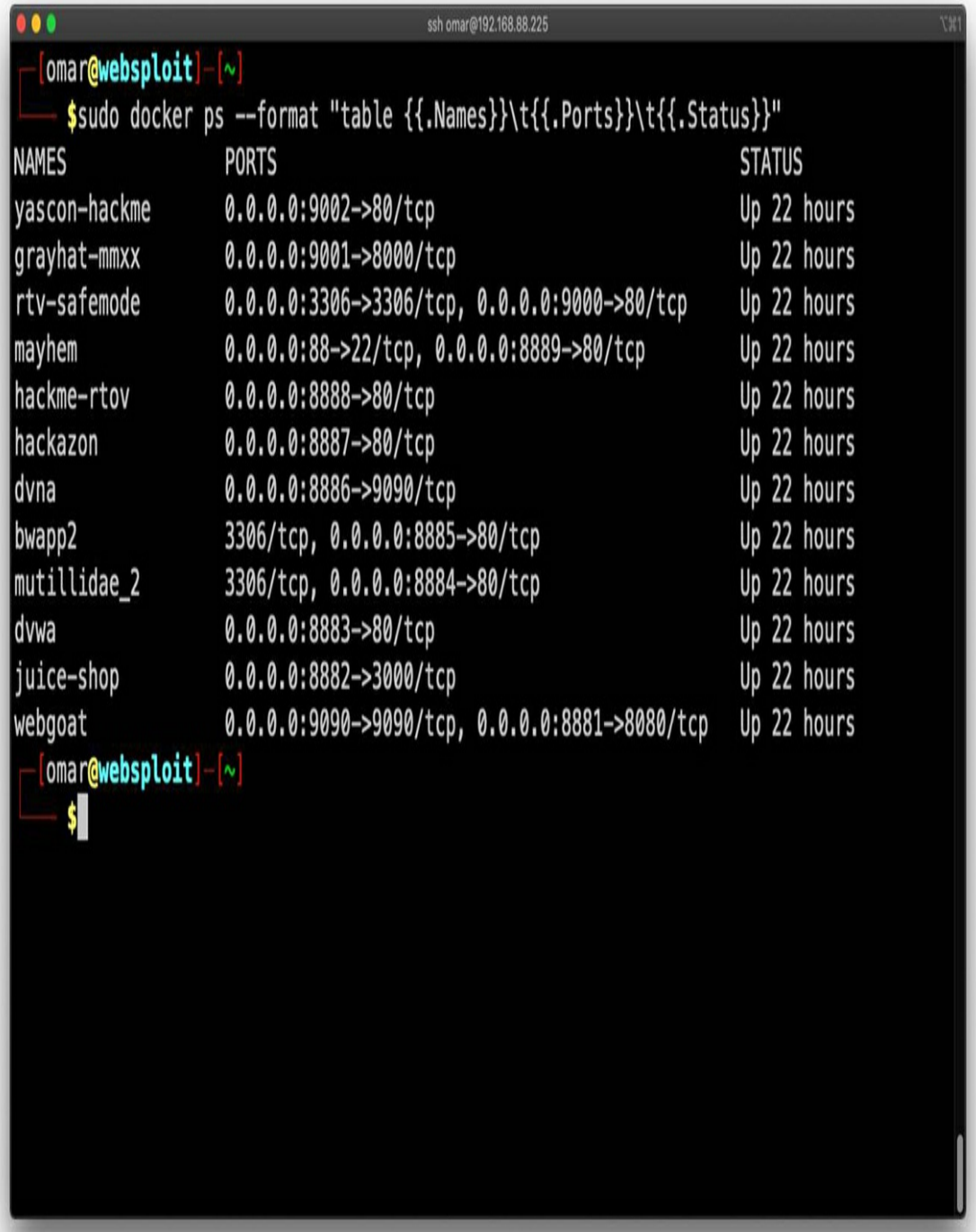


Figure 10-5 Output of the **docker ps** Command

You can use a public, cloud provider, or private Docker image repository.

Docker's public image repository is called Docker Hub (https://hub.docker.com). You can find images by going to the Docker Hub website or by using the **docker search** command, as demonstrated in Figure 10-6.

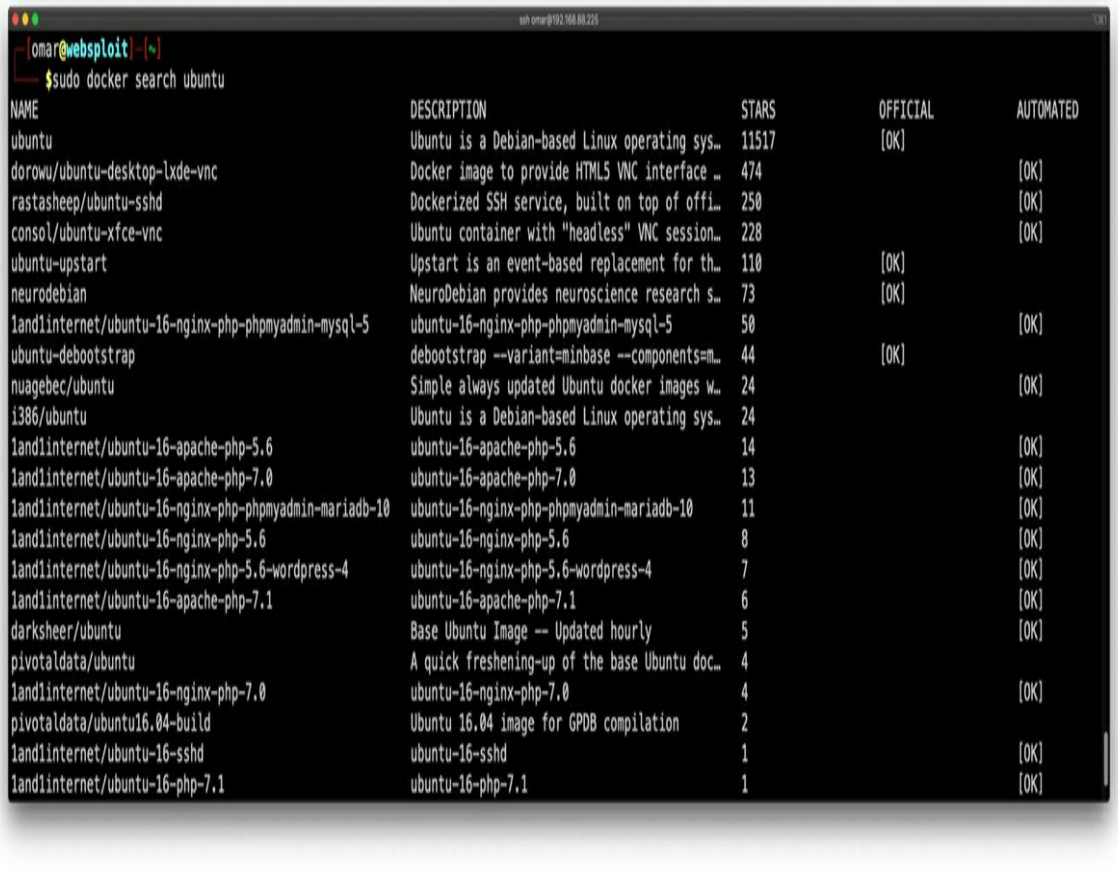


Figure 10-6 Output of the **docker search** Command

In Figure 10-6, the user searches for a container image that matches the **ubuntu** keyword.

Tip

You can practice and deploy your first container by using Katacoda (an interactive system that allows you to learn many different technologies, including Docker, Kubernetes, Git, TensorFlow, and many others). You can access Katacoda at www.katacoda.com. There are numerous interactive scenarios

provided by Katacoda. For instance, you can use the “Deploying Your First Docker Container” scenario to learn (hands-on) Docker: www.katacoda.com/courses/docker/deploying-first-container.

In larger environments, you will not deploy and orchestrate Docker containers in a manual way. You will want to automate as much as possible. This is where Kubernetes comes into play. Kubernetes (often referred to as k8s) automates the distribution, scheduling, and orchestration of application containers across a cluster.

The following are the Kubernetes components:

- **Control Plane:** This component coordinates all the activities in your cluster (scheduling, scaling, and deploying applications).
- **Node:** This VM or physical server acts as a worker machine in a Kubernetes cluster.
- **Pod:** This group of one or more containers provides shared storage and networking, including a specification for how to run the containers. Each pod has an IP address, and it is expected to be able to reach all other pods within the environment.

There have been multiple technologies and solutions to manage, deploy, and orchestrate containers in the industry. The following are the most popular:

- **Kubernetes:** One of the most popular container orchestration and management frameworks, originally developed by Google, Kubernetes is a platform for creating, deploying, and managing distributed applications. You can download Kubernetes and access its documentation at https://kubernetes.io.
- **Nomad:** This container management and orchestration platform is by HashCorp. You can download and obtain detailed information about it at www.nomadproject.io.
- **Apache Mesos:** This distributed Linux kernel provides native support for launching containers with Docker and AppC images. You can download Apache Mesos and access its documentation at https://mesos.apache.org.

- **Docker Swarm:** This container cluster management and orchestration system is integrated with the Docker Engine. You can access the Docker Swarm documentation at https://docs.docker.com/engine/swarm.

Microservices and APIs

The term ***microservices*** describes how applications can be deployed as a collection of services that are highly maintainable and testable and independently deployable. Microservices are organized around business capabilities and enable the rapid, frequent, and reliable delivery of large, complex applications. Most microservices developed and deployed by organizations are based on containers.

Application programming interfaces (APIs), such as RESTful APIs, are the frameworks through which developers can interact with an application. Microservices use APIs to communicate between themselves. These APIs can be used for inter-microservice communication, and they can also be used to expose data and application functionality to third-party systems.

Microservices and APIs need to be secured. Traditional segmentation strategies do not work well in these virtual and containerized environments. The ability to enforce network segmentation in container and VM environments is what people call *micro-segmentation*. Micro-segmentation is at the VM level or between containers regardless of a VLAN or a subnet. Micro-segmentation solutions need to be "application aware." This means that the segmentation process starts and ends with the application itself.

Most micro-segmentation environments apply a zero-trust model. This model dictates that users cannot talk to applications and that applications cannot talk to other applications unless a defined set of policies permits them to do so.

Infrastructure as Code

Modern applications and deployments (on-premises and in the cloud) require scalability and integration with numerous systems, APIs, and network components. This complexity introduces risks, including network

configuration errors that can cause significant downtime and network security challenges. Subsequently, networking functions such as routing, optimization, and security have also changed. The next generation of hardware and software components in enterprise networks must support both the rapid introduction and rapid evolution of new technologies and solutions. Network infrastructure solutions must keep pace with the business environment and support modern capabilities that help drive simplification within the network.

These elements have fueled the creation of ***software-defined networking (SDN)***. SDN was originally created to decouple control from the forwarding functions in networking equipment. This is done to use software to centrally manage and "program" the hardware and virtual networking appliances to perform forwarding. Organizations are adopting a framework often referred to as ***infrastructure as code***, which is the process of managing and provisioning computer data centers through machine-readable definition files rather than physical hardware configuration or interactive configuration tools.

In traditional networking, three different "planes," or elements, allow network devices to operate: the management, control, and data planes. The control plane has always been separated from the data plane. There was no central brain (or controller) that controlled the configuration and forwarding. Routers, switches, and firewalls were managed by the command-line interface (CLI), graphical user interfaces (GUIs), and custom Tcl scripts.

Note

Tcl is a high-level, general-purpose, interpreted, dynamic programming language. It was designed with the goal of being very simple but powerful. Tcl casts everything into the mold of a command, even programming constructs like variable assignment and procedure definition.

For instance, the firewalls were managed by standalone web portals, while the routers were managed by the CLI. SDN introduced the notion of a *centralized controller*. The SDN controller has a global view of the network, and it uses a common management protocol to configure the network

infrastructure devices.

In the topology shown in Figure 10-7, an SDN controller is used to manage a network infrastructure in a data center.

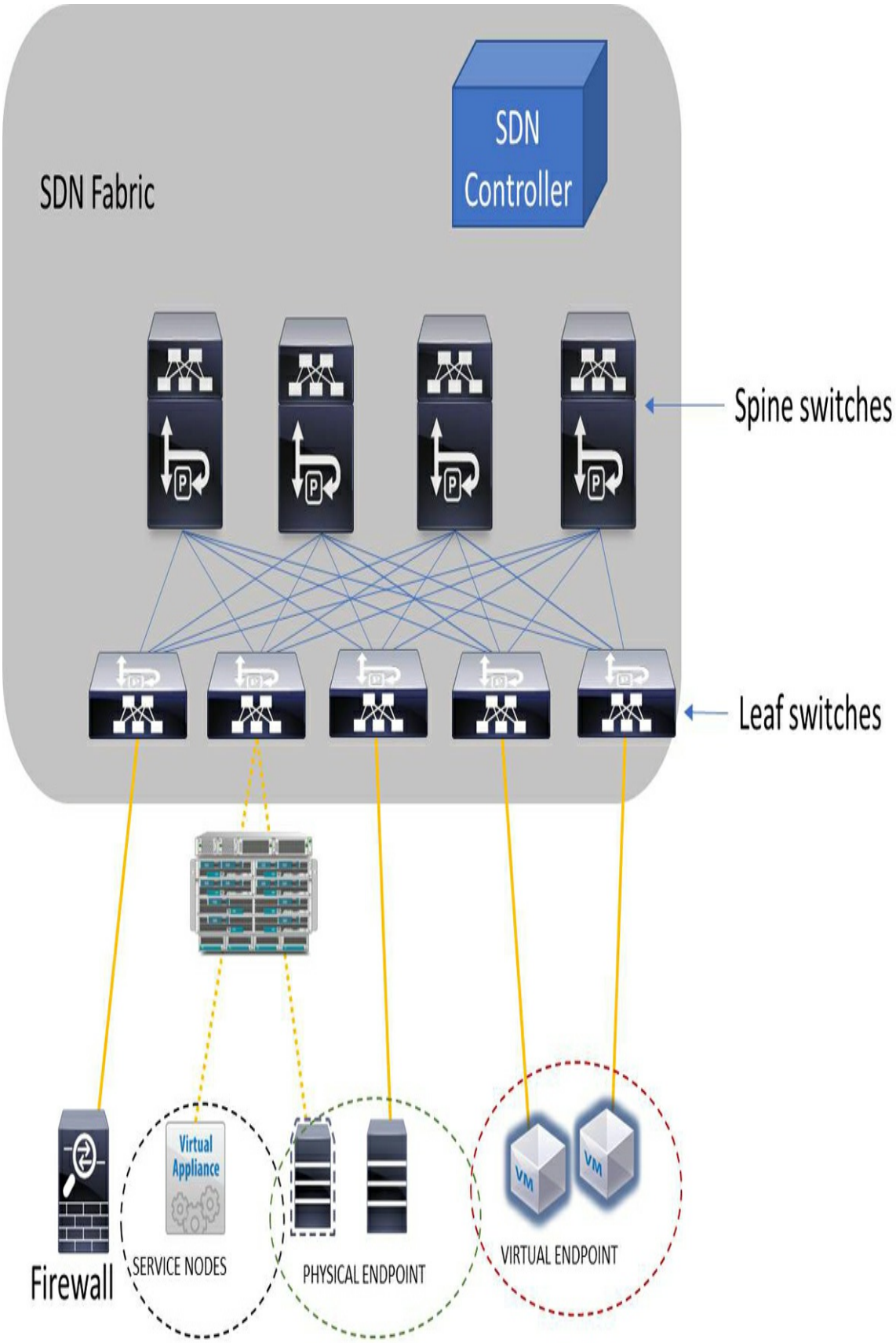

SDN Fabric
SDN Controller
Spine switches
Leaf switches
Virtual Appliance
VM
VM
Firewall
SERVICE NODES
PHYSICAL ENDPOINT
VIRTUAL ENDPOINT

Figure 10-7 A High-Level SDN Implementation

The SDN controller can also calculate reachability information from many systems in the network, and it pushes a set of flows inside the switches. The hardware uses the flows to do the forwarding. Here, you can see a clear transition from a distributed "semi-intelligent brain" approach to a "central and intelligent brain" approach.

Tip

An example of an open-source implementation of SDN controllers is the Open vSwitch (OVS) project using the OVS Database (OVSDB) management protocol and the OpenFlow protocol. Another example is the Cisco Application Policy Infrastructure Controller (Cisco APIC). Cisco APIC is the main architectural component and the brain of the Cisco Application Centric Infrastructure (ACI) solution. OpenDaylight (ODL) is another popular open-source project that is focused on the enhancement of SDN controllers to provide network services across multiple vendors. OpenDaylight participants also interact with the OpenStack Neutron project and attempt to solve the existing inefficiencies. OpenDaylight interacts with Neutron via a northbound interface and manages multiple interfaces southbound, including the Open vSwitch Database Management Protocol (OVSDB) and OpenFlow.

SDN changed a few things in the management, control, and data planes. However, the big change was in the control and data planes in software-based switches and routers (including virtual switches inside hypervisors). For instance, the Open vSwitch project started some of these changes across the industry.

SDN provides numerous benefits in the area of the management plane. These benefits are in both physical switches and virtual switches. SDN is now widely adopted in data centers.

Another concept introduced a few years ago is ***software-defined visibility (SDV)***. SDV is a concept similar to SDN but is designed and optimized to

provide intelligent visibility of what's happening across an organization network or in the cloud. Typically, RESTful APIs are used in an SDV environment to orchestrate and automate security operations and monitor tasks across the organization.

Serverless Architecture

Another popular architecture is the ***serverless architecture***. Be aware that *serverless* does not mean that you do not need a "server" somewhere. Instead, it means that you will be using cloud platforms to host and/or to develop your code. For example, you might have a serverless app that is distributed in a cloud provider like AWS, Azure, or Google Cloud Platform.

Serverless is a cloud computing execution model where the cloud provider (AWS, Azure, Google Cloud, and so on) dynamically manages the allocation and provisioning of servers. Serverless applications run in stateless containers that are ephemeral and event-triggered (fully managed by the cloud provider). AWS Lambda is one of the most popular serverless architectures in the industry. Figure 10-8 shows a "function" or application in AWS Lambda.

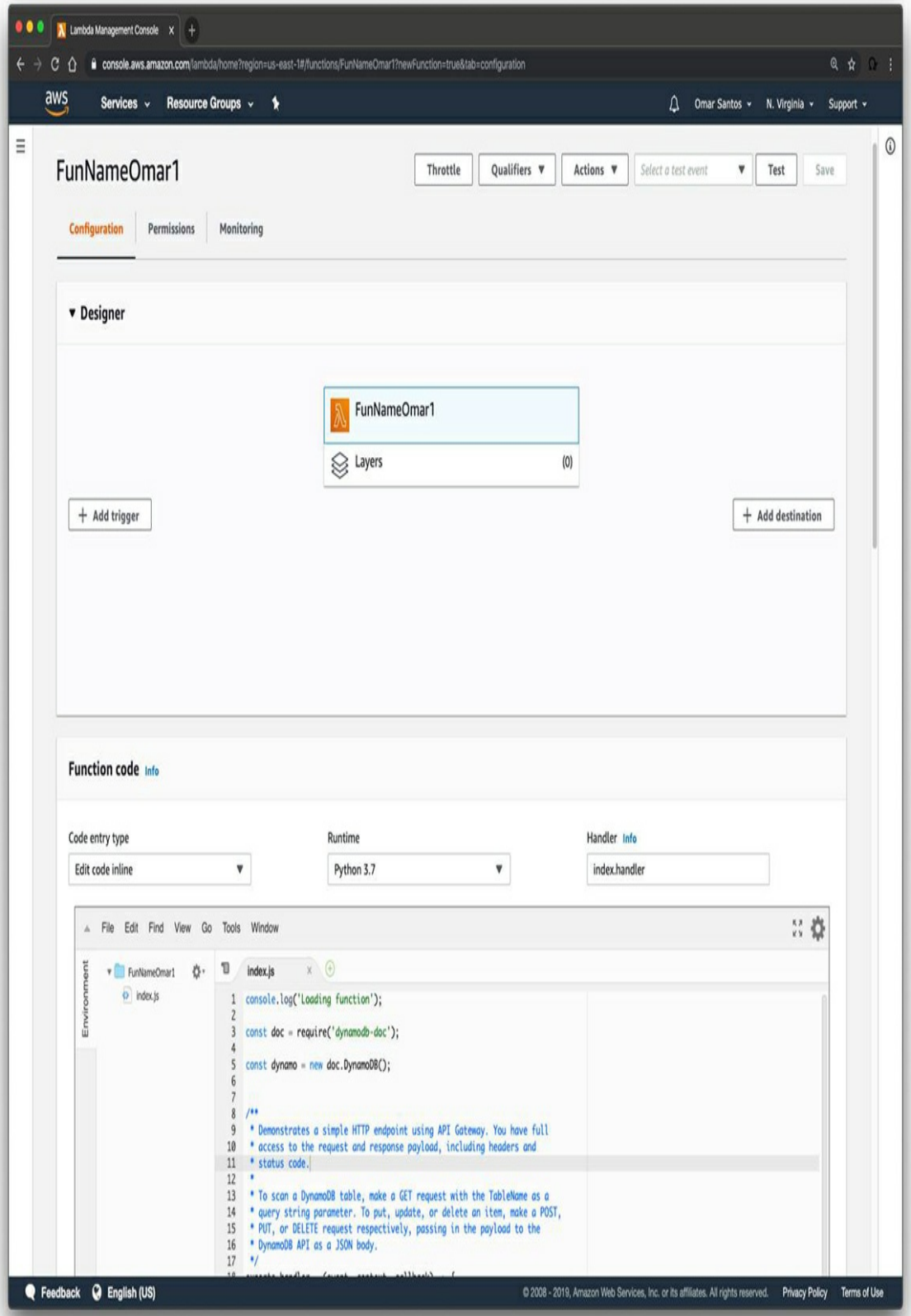

Lambda Management Console
console.aws.amazon.com/lambda/home?region=us-east-1#/functions/FunNameOmar1?newFunction=true&tab=configuration
Services
Resource Groups
Omar Santos
N. Virginia
Support
FunNameOmar1
Throttle
Qualifiers
Actions
Select a test event
Test
Save
Configuration
Permissions
Monitoring
Designer
FunNameOmar1
Layers
(0)
Add trigger
Add destination
Function code Info
Code entry type
Edit code inline
Runtime
Python 3.7
Handler Info
index.handler
File Edit Find View Go Tools Window
Environment
FunNameOmar1
index.js
index.js
1 console.log('Loading function');
2
3 const doc = require('dynamodb-doc');
4
5 const dynamo = new doc.DynamoDB();
6
7
8 /**
9 * Demonstrates a simple HTTP endpoint using API Gateway. You have full
10 * access to the request and response payload, including headers and
11 * status code.
12 *
13 * To scan a DynamoDB table, make a GET request with the TableName as a
14 * query string parameter. To put, update, or delete an item, make a POST,
15 * PUT, or DELETE request respectively, passing in the payload to the
16 * DynamoDB API as a JSON body.
17 */
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Figure 10-8 AWS Lambda

In AWS Lambda, you run code without provisioning or managing servers, and you pay only for the compute time you consume. When you upload your code, Lambda takes care of everything required to run and scale your application (offering high availability and redundancy).

Figure 10-9 summarizes the evolution of computing from physical servers to virtual machines, containers, and serverless solutions. Virtual machines and containers could be deployed on-premises or in the cloud.

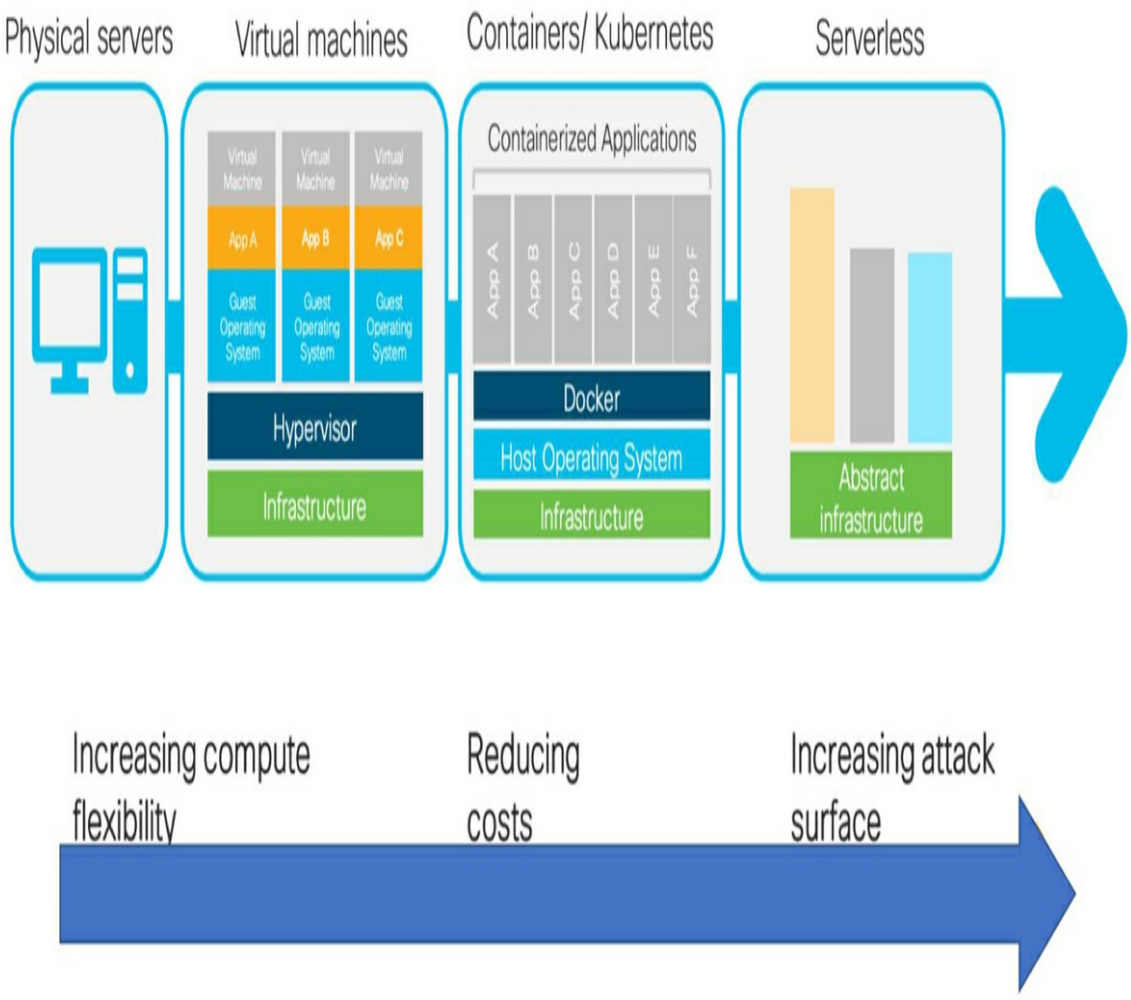


Figure 10-9 The Evolution of Computing from Physical Servers to Serverless

Services Integration

Cloud ***services integration*** is a series of tools and technologies used to connect different systems, applications, code repositories, and physical or virtual network infrastructure to allow real-time exchange of data and processes. Subsequently, integrated cloud services can be accessed by multiple systems, the Internet, or in the case of a private cloud, over a private network.

One of the main goals of cloud services integration is to break down data silos, increase visibility, and improve business processes. These services integrations are crucial for most organizations needing to share data among cloud-based applications. Cloud services integration has grown in popularity over the years as the use of IaaS, PaaS, and SaaS solutions continues to increase.

Resource Policies

Resource policies involve creating, assigning, and managing rules over the cloud resources that systems (virtual machines, containers, and so on) or applications use. Resource policies are also often created to make sure that resources remain compliant with corporate standards and service-level agreements (SLAs). Cloud service providers offer different solutions (such as the Microsoft Azure Policy) that help you maintain good cloud usage governance by evaluating deployed resources and scanning for those not compliant with the policies that have been defined.

Let’s look at a resource policy example. You could create a policy to allow only a certain size of virtual machines in the cloud environment. Once that policy is implemented, it would be enforced when you’re creating and updating resources as well as over existing resources. Another example is a policy that can contain conditions and rules that determine if a storage

account being deployed is within a certain set of criteria (size, how it is shared, and so on). Such a policy could deny all storage accounts that do not adhere to the defined set of rules.

Transit Gateway

A ***transit gateway*** is a system that can be used to interconnect a virtual private cloud (VPC) and on-premises networks. Transit gateways can also be used to connect cloud-hosted applications to software-defined wide-area network (SD-WAN) devices.

> **Note**
>
> SD-WAN is a solution that simplifies the management of a wide-area network (WAN) implementation by decoupling network infrastructure devices from its control mechanism using software.

Figure 10-10 illustrates a transit gateway implementation. On-premises network infrastructure and systems are connected to the transit gateway using a VPN tunnel (typically using IPsec). The transit gateway then connects those resources to different VPCs.

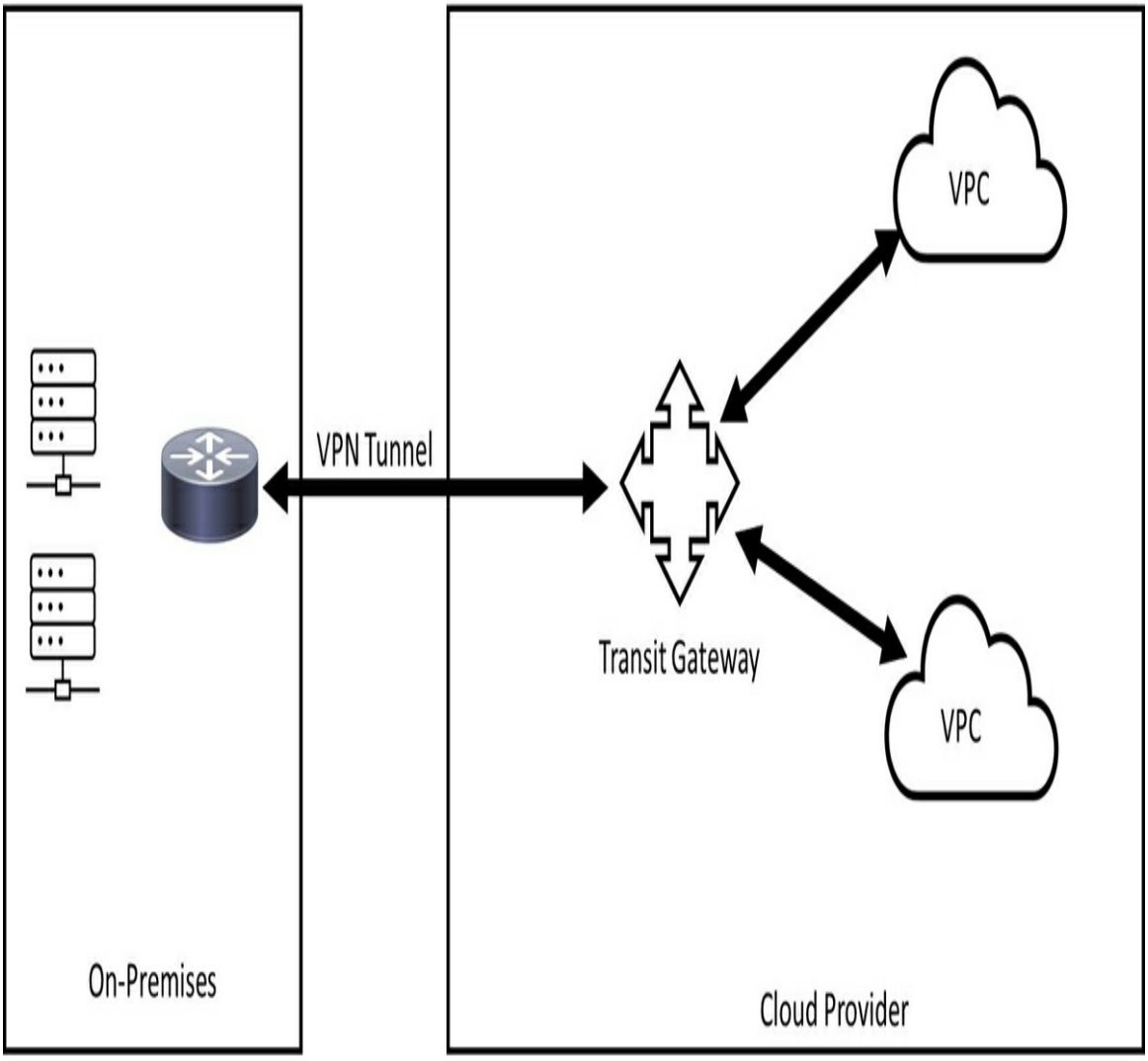


Figure 10-10 A Transit Gateway

Virtual Machine (VM) Sprawl Avoidance and VM Escape Protection

Virtualization is a solution that allows you to create IT services using hardware resources that can be shared among different installations of operating systems and applications. The individual virtual systems that run different operating systems and applications are called *virtual machines* (VMs). The hypervisor is the computer software that is used to create and run VMs. Organizations have used virtualization technologies for decades. There

have been a few challenges in virtualized environments, however. One of those issues is VM sprawl. In the following sections, you learn the challenges of VM sprawl and how to potentially avoid it. You also learn about VM escape attacks and how to protect your organization against them.

Understanding and Avoiding VM Sprawl

Regardless of the size of your organization, you may find yourself in a situation that too many virtual machines (or physical servers for that matter) may be deployed, which results in unnecessary complexity and inefficiency when those resources aren't used appropriately or are no longer needed. This unnecessary complexity also introduces security risks. ***VM sprawl*** (otherwise known as virtualization sprawl) occurs when an organization can no longer effectively control and manage all the VMs on a network or in the cloud.

VM sprawl can happen with rapidly growing networks when numerous VMs are set up by different business units for disperse applications—especially when you create temporary VMs that do not get purged. For instance, your developers or department might set up VMs as part of a testing environment; however, if the VMs aren't disposed properly when they are no longer needed, your IT department can end up with too much to manage.

VM sprawl can impact your organization in a few different ways:

- Cost
- Management complexity and overhead
- Introduction of additional security risks
- Unnecessary disk space, CPU, and memory consumption
- Potential data protection challenges

One of the best ways to avoid VM sprawl (VM sprawl avoidance) is to have a robust, automated solution to perform audits of the VMs. This automated audit will allow you to identify which VMs are no longer needed and delete them. Additionally, you should have a good process to clean up orphaned VM snapshots and end unnecessary backups or replication of VMs no longer

needed. It is also very important to use a good naming convention so that you can keep track of these VMs in your infrastructure or in the cloud.

Protecting Against VM Escape Attacks

In a virtualized environment, the hypervisor controls all aspects of VM operation. Securing a hypervisor is crucial but also fairly complex. There is an attack known as *hyper-jacking*. In this type of assault, an attacker or malware compromises one VM and then attacks the hypervisor. When a guest VM performs this type of attack, it is often called a ***VM escape attack***. In a VM escape attack, the guest VM breaks out of its isolated environment and attacks the hypervisor or could compromise other VMs hosted and controlled by the hypervisor. Figure 10-11 illustrates a VM escape attack.

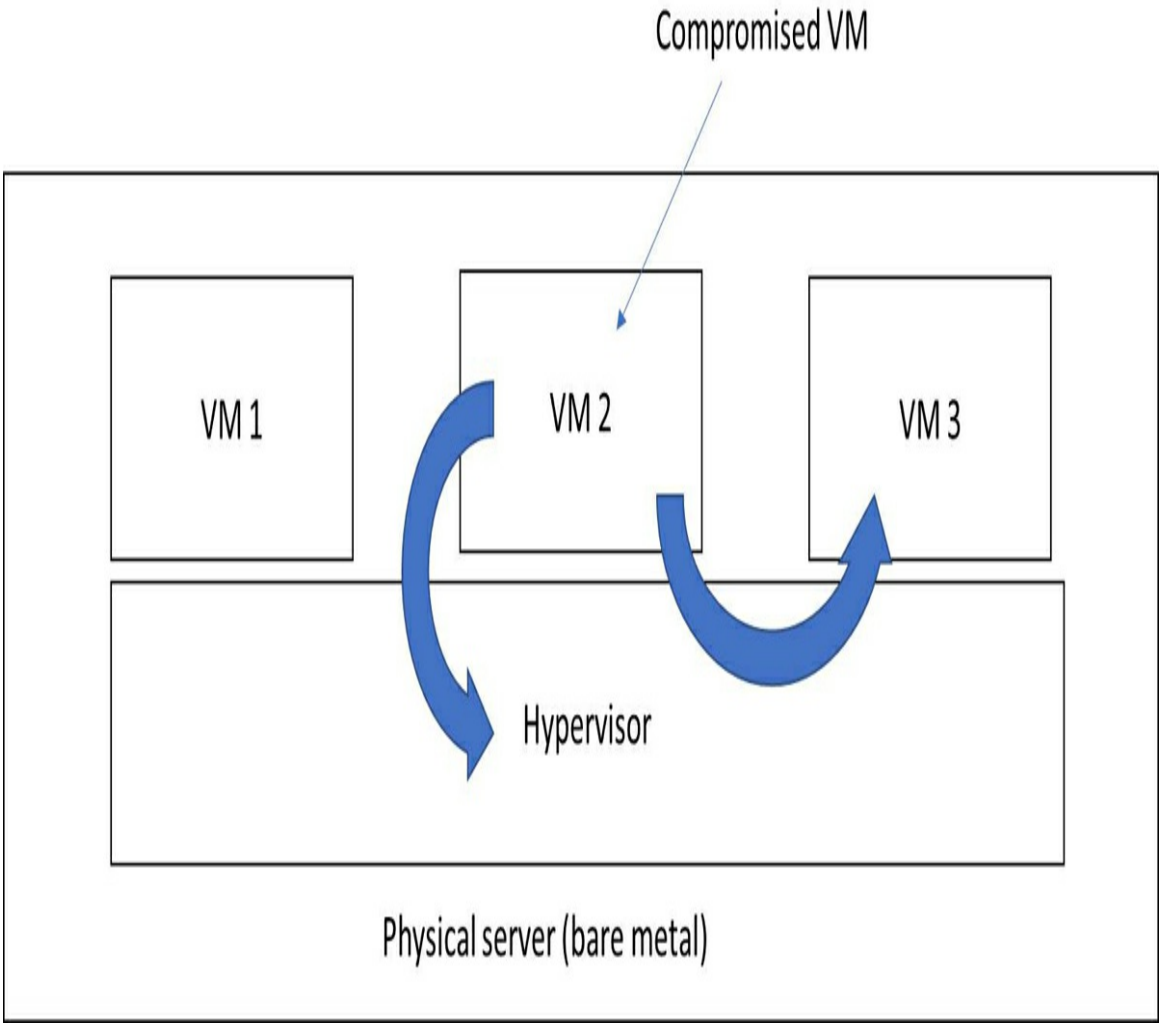


Figure 10-11 VM Escape Attack

Consider the following to mitigate the risk of VM escape attacks:

- Use a Type 1 hypervisor versus a Type 2 hypervisor to reduce the attack surface. A Type 1 hypervisor runs on physical servers or “bare metal,” and Type 2 runs on top of an operating system. Type 1 hypervisors have a smaller footprint and reduced complexity.
- Harden the hypervisor’s configuration per the recommendation of the vendor. For example, you may disable memory sharing between VMs running within the same hypervisor hosts.
- Disconnect unused external physical hardware devices (such as USB drives).

- Disable clipboard or file-sharing services.
- Perform self-integrity checks at boot-up to verify whether or not the hypervisor has been compromised using technologies like the Intel Trusted Platform Module/Trusted Execution Technology.
- Monitor and analyze hypervisor logs on an ongoing basis and look for indicators of compromise.
- Keep up with your hypervisor vendor's security advisories and apply security updates immediately after they are disclosed.
- Implement identity and access control across all tools, automated scripts, and applications using the hypervisor APIs.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 10-2 lists a reference of these key topics and the page number on which each is found.

Table 10-2 Key Topics for Chapter 10

Key Topic Element	Description	Page Number
List	Listing the cloud computing service categories	
Tip	Surveying NIST's special publication (SP) 800-145	
List	Listing the different types of clouds used by organizations	
Paragraph	Defining edge computing	
Figure 10-1	Edge computing	
Paragraph	Defining fog computing	
Paragraph	Understanding what thin clients are	
Figure 10-2	Thin client accessing a virtualized desktop	
Paragraph	Understanding what containers are	
Figure 10-3	How container images work	
Paragraph	Defining microservices	
Figure 10-7	A high-level example of an SDN implementation	
Paragraph	Understanding cloud services integration	
Paragraph	Defining resource policies	

Paragraph	Understanding what a transit gateway is	
Figure 10-10	An example of a transit gateway	
Section	Understanding and Avoiding VM Sprawl	
Paragraph	Understanding what a VM escape attack is	
Figure 10-11	VM escape attack	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

cloud computing

software as a service (SaaS)

infrastructure as a service (IaaS)

platform as a service (PaaS)

anything as a service (XaaS)

public cloud

private cloud

hybrid cloud

community cloud

cloud service provider (CSP)

managed service provider (MSP)

managed security service providers (MSSPs)

off-premises

on-premises

edge computing

fog computing

Thin clients

containers

microservices

application programming interfaces (APIs)

software-defined networking (SDN)

infrastructure as code

software-defined visibility (SDV)

serverless architecture

services integration

resource policies

transit gateway

virtualization

VM sprawl

VM escape attack

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What system can be used to interconnect a virtual private cloud (VPC) and on-premises networks?
2. _____ are used in the process of creating, assigning, and managing rules over the cloud resources that systems (virtual machines, containers, and so on) or applications use.
3. What is a series of tools and technologies used to connect different systems, applications, code repositories, and physical or virtual network infrastructure to allow the real-time exchange of data and processes?
4. AWS Lambda is an example of which type of cloud service architecture?
5. OpenDaylight (ODL) is an example of a(n) _____ controller.

Chapter 11. Summarizing Secure Application Development, Deployment, and Automation Concepts

This chapter covers the following topics related to Objective 2.3 (Summarize secure application development, deployment, and automation concepts) of the CompTIA Security+ SY0-601 certification exam:

- Environment
  - Development
  - Test
  - Staging
  - Production
  - Quality assurance (QA)
- Provisioning and deprovisioning
- Integrity measurement
- Secure coding techniques
  - Normalization
  - Stored procedures
  - Obfuscation/camouflage
  - Code reuse/dead code
  - Server-side vs. client-side execution and validation
  - Memory management
  - Use of third-party libraries and software development kits (SDKs)
  - Data exposure

- Open Web Application Security Project (OWASP)
- Software diversity
    - Compiler
    - Binary
- Automation/scripting
    - Automated courses of action
    - Continuous monitoring
    - Continuous validation
    - Continuous integration
    - Continuous delivery
    - Continuous deployment
- Elasticity
- Scalability
- Version control

This chapter provides an overview of different software development environments and methodologies. You then learn how organizations provision and deprovision applications in modern environments. You also learn about software integrity measurement, different secure coding techniques, software diversity, and how automation and orchestration have a direct effect in elasticity and scalability.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz enables you to assess whether you should read this entire chapter thoroughly or jump to the "Exam Preparation Tasks" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. Table 11-1 lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Review Questions."

Table 11-1 "Do I Know This Already?" Section-to-Question Mapping

Foundation Topics Section	Questions
Software Development Environments and	1
Methodologies	
Application Provisioning and Deprovisioning	2
Software Integrity Measurement	3
Secure Coding Techniques	4–6
Open Web Application Security Project (OWASP)	7
Software Diversity	8
Automation/Scripting	9
Elasticity and Scalability	10

Caution

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

1. Which of the following is a software and hardware development and project management methodology that has at least five to seven phases that follow in strict linear order?

a. Waterfall

b. Agile

c. DevOps

d. SDLC

2. Which of the following is a benefit when you use log aggregation tools to maintain and analyze logs of every element that goes into the provisioning of applications?

a. The ability to scale horizontally

b. The ability to respond quickly and deprovision the application in the event that something wrong

c. The ability to design an elastic infrastructure

d. None of these answers are correct.

3. Which of the following elements can help software (code) integrity?

a. Unit testing

b. Integration testing

c. Identifying a code integrity manager

d. All of these answers are correct.

4. Which process includes identifying assets to the system or application, uncovering vulnerabilities, identifying threats, documenting threats, and rating those threats according to their potential impact?

a. SECOPS

b. Principle of least privilege

c. Threat modeling

d. None of these answers are correct.

5. Which of the following are important security principles that should be incorporated into the SDLC?

a. Input validation

b. Principle of least privilege

c. Failing securely

d. All of these answers are correct.

6. Which of the following might include syntax errors in the code and type-checking errors?

a. Misconfigured VMs

b. Unpatched applications and operating systems

c. Misconfigured storage buckets

d. Compile-time errors

7. Which of the following are top web application security risks?

a. Broken Access Control

b. XML External Entities (XXE)

c. Cross-Site Scripting (XSS)

d. All of these answers are correct.

8. Which of the following terms is often used when a compiler is modified to generate variants of a binary (target application) that operates in the same way when processing benign input but may operate in a different manner when given malicious input?

a. ASLR

b. Random forest

c. Software diversity

d. None of these answers are correct.

9. Which of the following is a software development practice where programmers merge code changes in a central repository multiple times a day?

a. Continuous integration

b. DevSecOps

c. Waterfall

d. All of these answers are correct.

10. Which of the following is the ability of an underlying infrastructure to react to a sudden increase in demand by provisioning more resources in an automated way?

a. Load balancing

b. Using Kubernetes

c. Elasticity

d. All of these answers are correct.

Foundation Topics

Software Development Environments and Methodologies

Before we start to define what DevOps is, let's look at the history of development methodologies and the underlying ***software development environments***. Decades of lessons learned from software development, high-reliability organizations, manufacturing, high-trust management models, and others have evolved into the DevOps practices used today.

The traditional software development methodology is the waterfall model, which is a software and hardware development and project management methodology that has at least five to seven phases that follow in strict linear order. Each phase cannot start until the previous phase has been completed. Figure 11-1 illustrates the typical phases of the waterfall development methodology.

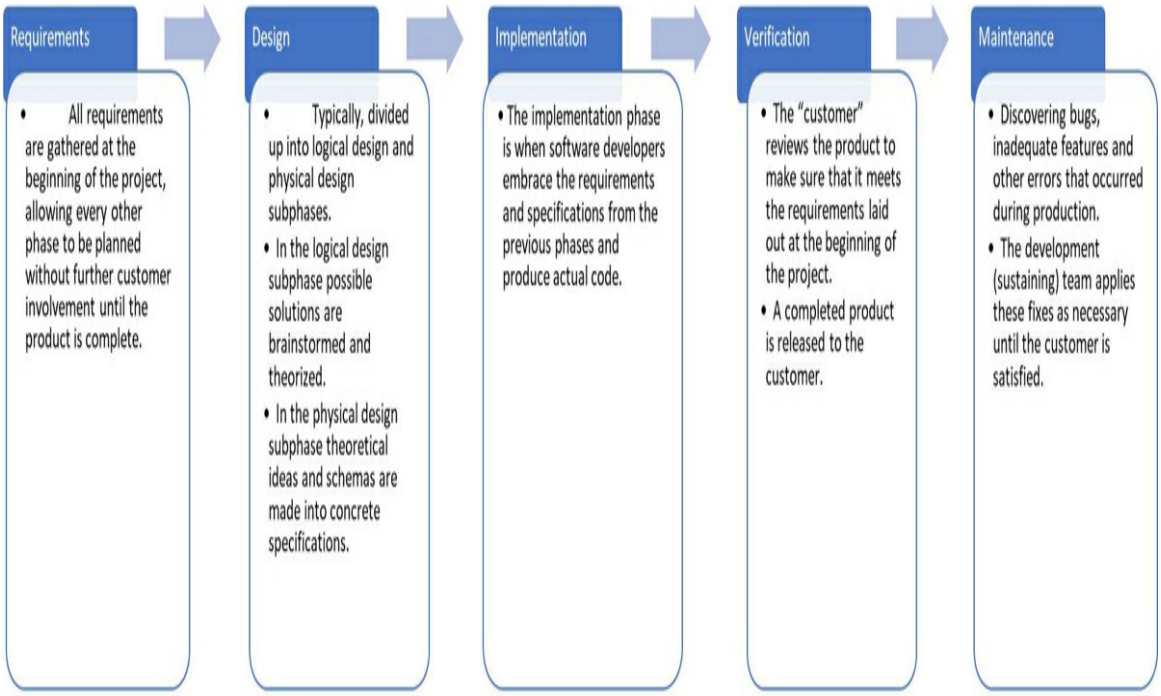


Figure 11-1 The Waterfall Development Methodology Phases

Organizations use the waterfall methodology for a few reasons. One of the main reasons is that project requirements are agreed upon from the beginning; consequently, planning and scheduling are simple and clear. With a fully laid-out project schedule, an accurate estimate can be given, including development project cost, resources, and deadlines. Another reason is that measuring progress is easy as the organization moves through the phases and hits the different milestones. End customers are not perpetually adding new requirements to the project, thus delaying production.

There also are several disadvantages to using the waterfall methodology. One

of the disadvantages is that it can be difficult for customers to enumerate and communicate all of their needs at the beginning of the project. If end customers are dissatisfied with the product in the verification phase, going back and designing the code again can be very costly. In the waterfall methodology, a linear project plan is rigid and lacks flexibility for adapting to unexpected events.

The next software development methodology is Agile. Agile is a software development and project management process in which a project is managed by breaking it up into several stages and involving constant collaboration with stakeholders and continuous improvement and iteration at every stage. The Agile methodology begins with end customers describing how the final product will be used and clearly articulating what problem it will solve. Once the coding begins, the respective team members cycle through a process of planning, executing, and evaluating. This process may allow the final deliverable to change in order to better fit the customers' needs. In an Agile environment, continuous collaboration is key. Clear and ongoing communication among team members and project stakeholders allows them to make fully informed decisions. The Agile methodology was originally developed in written form by 17 people in 2001, and it is documented at "The Manifesto for Agile Software Development" (https://agilemanifesto.org).

In Agile, the input to the development process is the creation of a business objective, concept, idea, or hypothesis. Then the work is added to a committed "backlog." From there, software development teams that follow the standard Agile or iterative process will transform that idea into "user stories" and some sort of feature specification. This specification is then implemented in code. The code is then checked in to a ***version control*** repository (for example, GitLab or GitHub), where each change is integrated and tested with the rest of the software system. In Agile, value is created only when services are running in production; consequently, the organization must ensure that it is not only delivering fast flow but also that deployments can be performed without causing chaos and disruptions, such as service outages, service impairments, or security or compliance failures. Figure 11-2 illustrates the general steps of the Agile methodology.

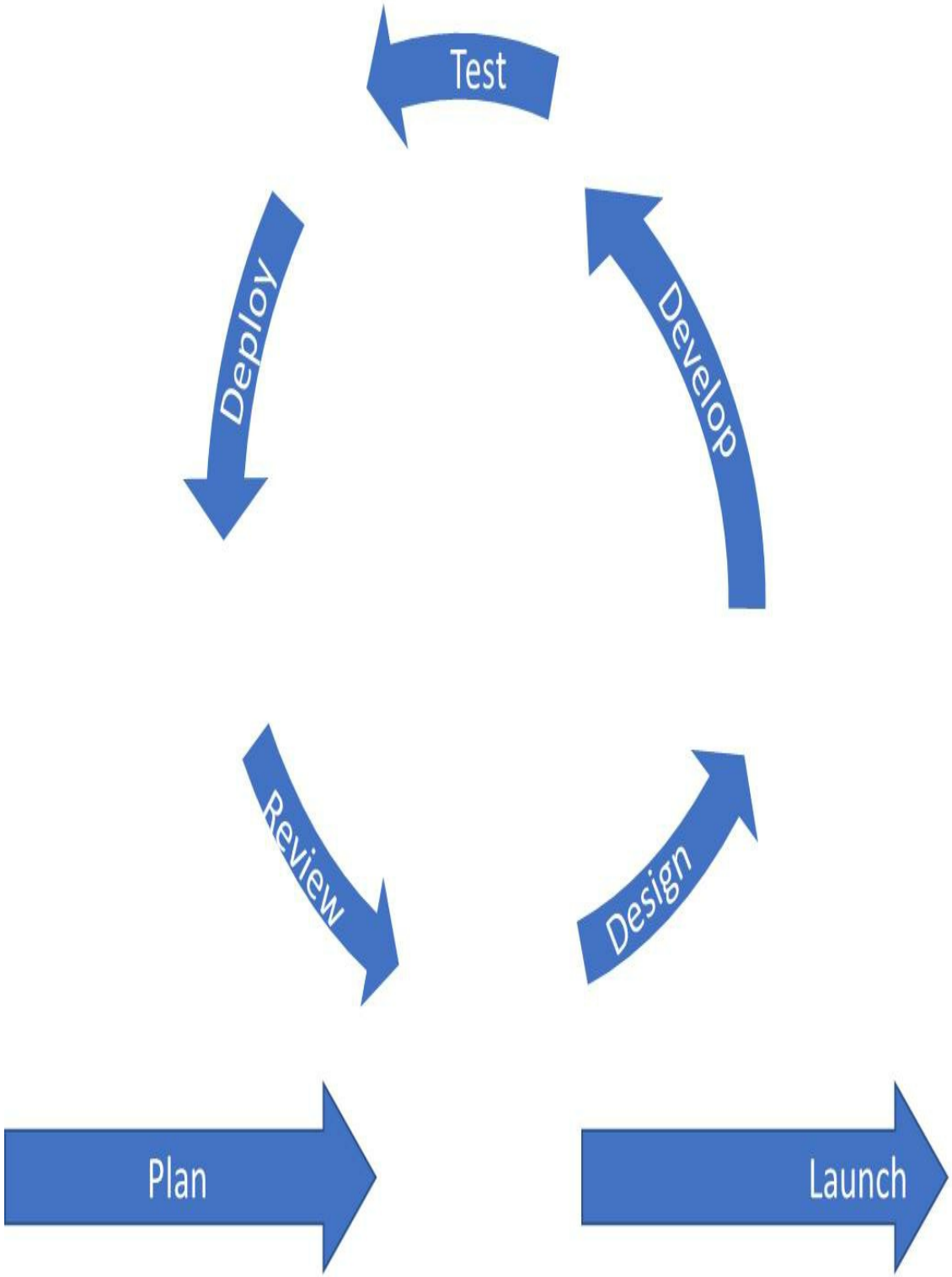

Test
Develop
Design
Launch
Review
Plan
Deploy

Figure 11-2 The Agile Methodology Phases

Many organizations adopt a concept related to Agile called Scrum. Scrum is a framework that helps organizations work together because it encourages teams to learn through experiences, self-organize while working on a solution, and reflect on their wins and losses to continuously improve. Scrum is used by software development teams; however, its principles and lessons can be applied to all kinds of teamwork. Scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.

Tip

Scrum.org provides a set of resources, certification, and training materials related to Scrum.

The increasing popularity of the Agile model led to DevOps and, subsequently, the practice of secure DevOps (or DevSecOps). DevOps is the outcome of many trusted principles—from software development, manufacturing, and leadership to the information technology value stream.

Regardless of the development methodology, most developers deal with different environments at some point. Each development environment has its own properties. The main environments are as follows:

- ***Development*:** In this environment you create your code on your computer or in the cloud. All of your commits and branches live here, along with those of your software development collaborators. Preliminary testing could happen in this environment. However, there is also a testing environment.
- ***Testing*:** This dedicated environment helps you make sure that your code does its job.
- ***Staging*:** This environment also allows you to test your code or application; however, it is as similar to the production environment as it

can be. The staging environment allows you to ensure that each component of your application still does its job with everything else going on around it. Systems and your application in the staging environment connect to as many services as they can without affecting the production environment. A lot of the hardcore quality assurance testing takes place at the staging environment. Remember, staging environments reduce the risk of introducing issues before solutions are deployed in production.

- ***Production*:** This is the environment where your code fulfills its destiny (where users access the final code after all of the updates and testing). When you hear people talk about making "code go live," this is the environment they're talking about.
- ***Quality assurance (QA)*:** This is the process and methods of making sure that your organization and software developers create good quality software. A good QA process should include standards and procedures that managers, developers, and administrators could use to review and audit your software development.

Application Provisioning and Deprovisioning

Traditionally, configuring IT infrastructure, opening firewall ports, provisioning certificates, and performing other similar tasks fell on the hands of the network engineer. However, in modern environments, a lot of this provisioning happens automatically. In Chapter 10, "Summarizing Virtualization and Cloud Computing Concepts," you learned about the concept of provisioning and configuring the network and IT infrastructure as code (IaC). In the past, the network engineer's job was to log in to each device and do the necessary configurations manually, and the entire process was laborious and painstaking. Furthermore, ***application provisioning*** is typically not a one-person job. Application provisioning requires the coordination and cooperation of several stakeholders from multiple teams (that is, developers, application owners, IT network administrators, and security teams).

With the use of modern applications and tools like Ansible and Terraform, software developers and network engineers are creating and provisioning

applications faster than ever. Using these solutions, you can provision an application in minutes rather than days or weeks.

Moreover, you can integrate with log aggregation tools to maintain and analyze logs of every element that goes into the provisioning. Therefore, you can respond quickly and ***deprovision*** the application in the event that something went wrong. Once you go back, you can check the logs and accurately find and fix the root cause of the error.

Traditionally, when an application needed to be fully retired or decommissioned, application deprovisioning was always a painful task. When you start the deprovisioning planning, you may not know how taking down one single application may have a direct effect on other applications that might have been integrated to it or that are using its data. Having a good automated process of logging, integration monitoring, and configuration will allow you to reduce the time needed to decommission the application and reduce the overall risk to the organization.

Software Integrity Measurement

Software integrity measurement (code integrity measurement) is the process of how to measure the source code's quality when it is passed on to the quality assessment (QA). This measurement is affected by how extensively the code was unit and integration tested. The purpose of code integrity measurement is for the developer to be sure that the code is written correctly. In short, code integrity checking helps organizations release better software with fewer bugs and security vulnerabilities.

Figure 11-3 includes a few elements that can help improve code integrity.

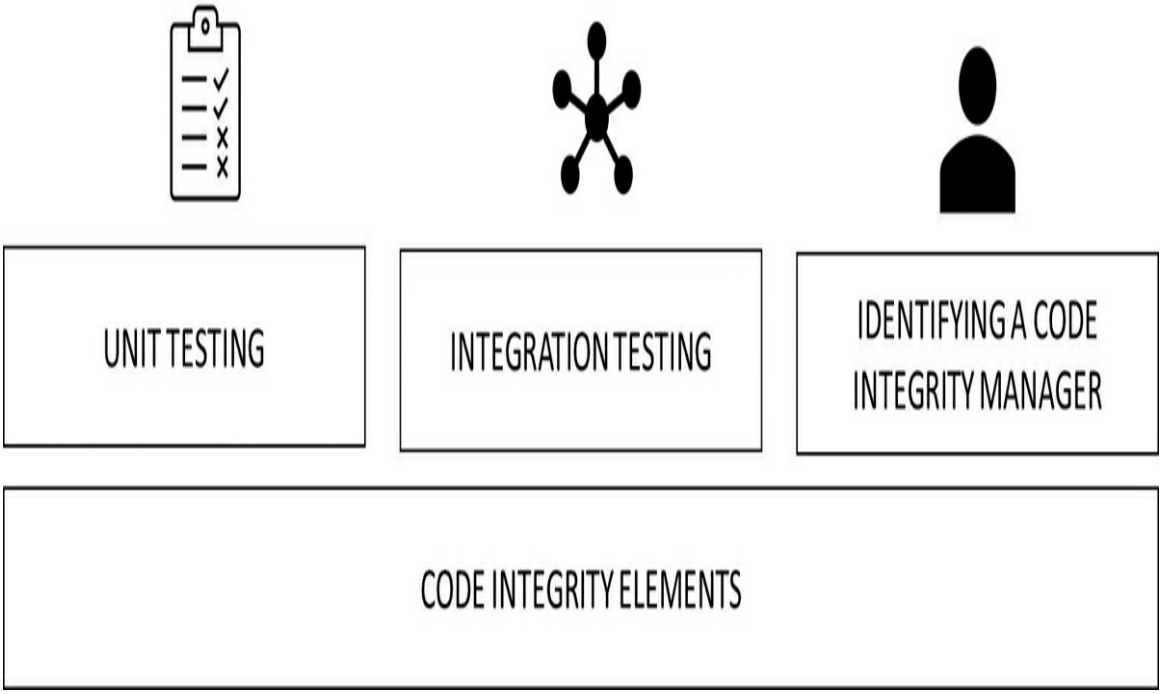


Figure 11-3 Elements That Help with Code Integrity

Figure 11-4 shows some of the advantages of code integrity checks and software integrity measurements.

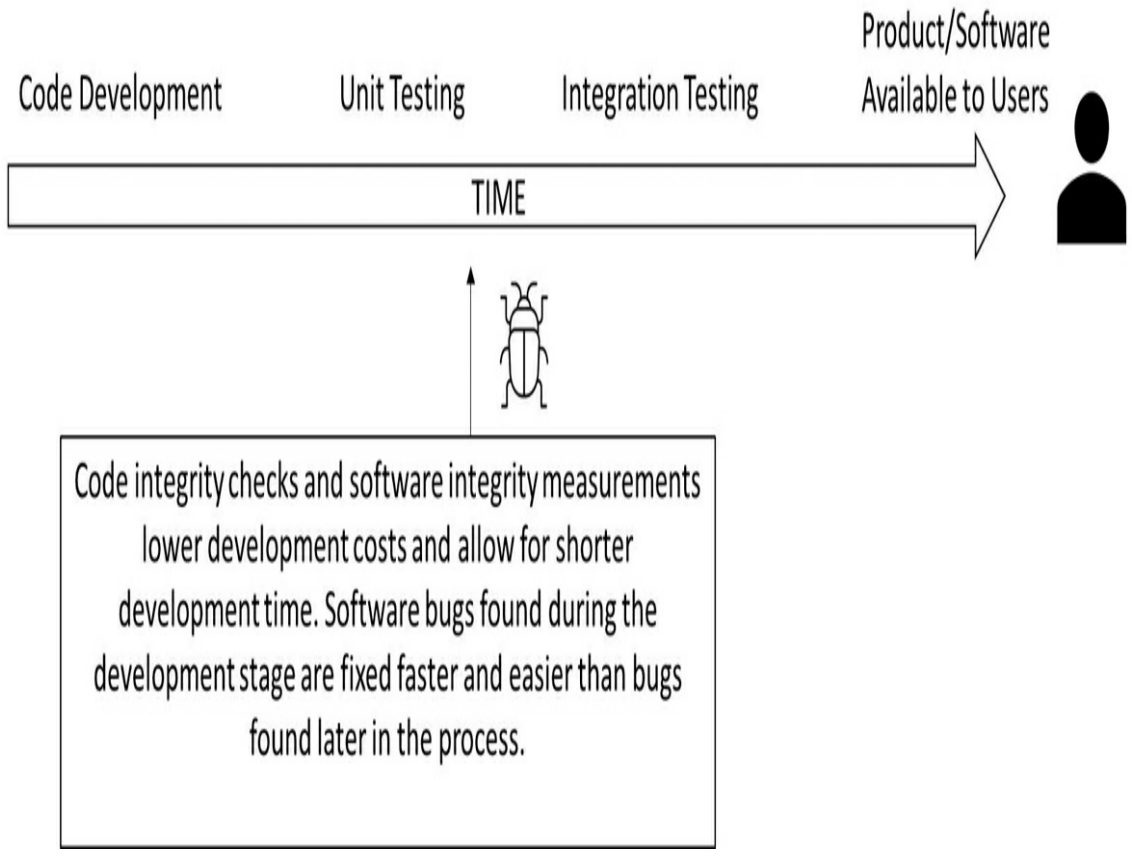


Figure 11-4 Advantages of Code Integrity Checks

Secure Coding Techniques

To properly develop a secure application, the developer has to scrutinize it at every turn, and from every angle, throughout the life of the project. Over time, this idea manifested itself into the concept known as the software development lifecycle (SDLC)—an organized process of planning, developing, testing, deploying, and maintaining systems and applications, and the various methodologies used to do so. A common SDLC model used by companies is the waterfall model, mentioned previously. In this model, the SDLC is broken down into several sequential phases. Here's an example of an SDLC's phases based on the waterfall model:

1. Planning and analysis. Goals are determined, needs are assessed, and

high-level planning is accomplished.

2. **Software/systems design.** The design of the system or application is defined and diagrammed in detail.
3. **Implementation.** The code for the project is written.
4. **Testing.** The system or application is checked thoroughly in a testing environment.
5. **Integration.** If multiple systems are involved, the application should be tested in conjunction with those systems.
6. **Deployment.** The system or application is put into production and is now available to end users.
7. **Maintenance.** Software is monitored and updated throughout the rest of its lifecycle. If there are many versions and configurations. version control is implemented to keep everything organized.

This is a basic example of the phases in an SDLC methodology. This one actually builds on the original waterfall model but is quite similar. However, it could be more or less complicated depending on the number of systems, the project goals, and the organization involved.

Note

The SDLC is also referred to as the *software development process*, systems *development lifecycle*, and application *development lifecycle*. CompTIA uses the term Software *Development Life Cycle* in its Security+ objectives acronym list, but be ready for slightly different terms based on the same concept.

While variations of the waterfall model are commonplace, an organization might opt to use a different model, such as the V-shaped model, which stresses more testing, or rapid application development (RAD), which puts more emphasis on process and less emphasis on planning. As previously mentioned, the Agile model allows you to break the software development process and work into small increments and is designed to be more adaptive

to change. It focuses on customer satisfaction, cooperation between developers and businesspeople, face-to-face conversation, simplicity, and quick adjustments to change.

The increasing popularity of the Agile model led to DevOps and, subsequently, the practice of secure DevOps. DevOps is a portmanteau of the terms *software development* and *information technology operations*. It emphasizes the collaboration of those two departments so that the coding, testing, and releasing of software can happen more efficiently—and hopefully, more securely. DevOps is similar to continuous delivery—which focuses on automation and quick execution—but from an organizational standpoint is broader and supports greater collaboration. A secure DevOps environment should include secure provisioning and deprovisioning of software, services, and infrastructure; security automation; continuous integration; baselining; infrastructure as code; and immutable systems. *Immutable* means unchanging over time. From a systems and infrastructure viewpoint, it means that software and services are replaced instead of being changed. Rapid, efficient deployment of new applications is at the core of DevOps, and it is one of the main tasks of all groups involved.

Core SDLC and DevOps Principles

The terms and models discussed in the previous section can be confusing at times because they are similar and because there are overlapping gray areas between them. So, if you are ever confused while working in the field, it's good to think back to the core fundamentals of security. From a larger perspective, you, as security administrator, and the programmer/systems developer should always keep the CIA concept in mind:

- **Maintaining confidentiality:** Allowing users access only to data to which they have permission
- **Preserving integrity:** Ensuring data is not tampered with or altered
- **Protecting availability:** Ensuring systems and data are accessible to authorized users when necessary

The CIA concepts are important when doing a secure code review, which can be defined as an in-depth code inspection procedure. Organizations often include it as part of the testing phase of the SDLC, but it is usually conducted

before other tests such as fuzzing or penetration tests, which are discussed later in this chapter.

In general, quality assurance policies and procedures should be implemented while developing and testing code. This implementation will vary from one organization to the next but generally includes procedures that have been developed by a team, a set of checks and balances, and a large amount of documentation. By checking the documentation within a project, a developer can save a lot of time, while keeping the project more secure.

From a larger perspective, an organization might implement modeling as part of its software quality assurance program. By modeling, or simulating, a system or application, the organization can test, verify, validate, and finally accredit it as acceptable for a specific purpose.

One secure and structured approach that organizations take is called threat modeling. Threat modeling enables you to prioritize threats to an application based on their potential impact. This modeling process includes identifying assets to the system or application, uncovering vulnerabilities, identifying threats, documenting threats, and rating those threats according to their potential impact. The more risk, the higher the rating. Threat modeling is often incorporated into the SDLC during the design, testing, and deployment phases.

Some other important security principles that should be incorporated into the SDLC include

- **Apply the principle of least privilege:** Applications should be coded and run in such a way as to maintain the principle of least privilege. Users should have access only to what they need. Processes should run with only the bare minimum access needed to complete their functions. However, this can be coupled with separation of privilege, where access to objects depends on more than one condition (for example, authentication plus an encrypted key).
- **Apply the principle of defense in depth:** The more security controls, the better. The layering of defense in secure coding may take the form of validation, encryption, auditing, special authentication techniques, and so on.
- **Ensure applications never trust user input:** Input should be validated

carefully using input validation techniques.

- **Minimize the attack surface area:** Every additional feature that a programmer adds to an application increases the size of the attack surface and increases risk. Unnecessary functions should be removed, and necessary functions should require authorization.
- **Establish secure defaults:** Out-of-the-box offerings should be as secure as possible. If possible, user password complexity and password aging default policies should be configured by the programmer, not the user. Permissions should default to no access and should be granted only as they are needed.
- **Provide for authenticity and integrity:** For example, when deploying applications and scripts, use *code signing* in the form of a cryptographic hash with verifiable checksum for validation. A digital signature will verify the author and/or the version of the code—that is, if the corresponding private key is secured properly.
- **Fail securely:** At times, applications will fail. How they fail determines their security. Failure exceptions might show the programming language that was used to build the application, or worse, lead to access holes. Error handling/exception handling code should be checked thoroughly so that a malicious user can't find out any additional information about the system. These error-handling methods are sometimes referred to technically as pseudocodes. For example, to handle a program exception, a properly written pseudocode will basically state (in spoken English): "If a program module crashes, then restart the program module."
- **Fix security issues correctly:** Once found, security vulnerabilities should be thoroughly tested, documented, and understood. Patches should be developed to fix the problem, but not cause other issues or application regression.

There are some other concepts to consider. First is ***obfuscation*** (or ***camouflage***), which is the complicating of source code to make it more difficult for people to understand. Code is obfuscated to conceal its purpose

in order to prevent tampering and/or reverse engineering. It is an example of security through obscurity. Other examples of security through obscurity include *code camouflaging* and *steganography*, both of which might be used to hide the true code being used. Another important concept is *code checking*, which involves limiting the reuse of code to that which has been approved for use, and removing dead code. It's also vital to incorporate good ***memory management*** techniques. Secure memory management and memory protection techniques include different ways to control memory access rights on a system to prevent a process from accessing memory that has not been allocated to it. Memory management best practices prevent memory-related vulnerabilities and potential malware within a process from affecting other processes or the underlying operating system. Secure memory protection should include all accesses to a specified area of memory, write accesses, or attempts to execute the contents of the memory area. Examples of memory protection techniques include address space layout randomization (ASLR) and executable space protection.

Finally, be very careful when using ***third-party libraries and software development kits (SDKs)***, and test them thoroughly before using them within a live application.

For the Security+ exam, the most important of the SDLC phases are maintenance and testing. In the maintenance phase, which doesn't end until the software is removed from all computers, an application needs to be updated accordingly, corrected when it fails, and constantly monitored. In the testing phase, a programmer (or team of programmers and other employees) checks for bugs and errors in a variety of ways. It's imperative that you know some of the vulnerabilities and attacks to a system or application and also how to fix them and protect against them. The best way to prevent these attacks is to test and review code.

Programming Testing Methods

Several testing methods and techniques can be implemented to seek out programming vulnerabilities and help prevent attacks from happening. Programmers have various ways to test their code, including system testing, input validation, and fuzzing. By using a combination of these testing techniques during the testing phase of the SDLC, there is a much higher

probability of a secure application as the end result.

In some security assessments and testing, knowledge of the system code—and programming knowledge in general—is not required. The tester does not know about the system's internal structure and is often given limited information about what the application or system is supposed to do. In this type of testing, one of the most common goals is to crash the program. If a user is able to crash the program by entering improper input, the programmer has probably neglected to thoroughly check the error-handling code and/or input validation.

On the other side of the spectrum, testers have in-depth architectural knowledge and are given detailed information about the design of the system. They are given login details, production documentation, and source code. System testers might use a combination of fuzzing (covered shortly), data flow testing, and other techniques such as stress testing, penetration testing, and sandboxes. *Stress testing* is usually done on real-time operating systems, mission-critical systems, and software, and checks whether they have the robustness and availability required by the organization. As you learned in Chapter 8, "Understanding the Techniques Used in Penetration Testing," a *penetration test* is a method of evaluating a system's security by simulating one or more attacks on that system. *Sandbox* is the term applied when a web script (or other code) runs in its own environment (often a virtual environment) for the express purpose of not interfering with other processes, often for testing. Sandboxing technology is frequently used to test unverified applications for malware, malignant code, and possible errors such as buffer overflows.

Compile-Time Errors vs. Runtime Errors

Programmers and developers need to test for potential compile-time errors and runtime errors. Compile time refers to the duration of time during which the statements written in any programming language are checked for errors. *Compile-time errors* might include syntax errors in the code and type-checking errors. A programmer can check these without actually running the program and instead check it in the compile stage when it is converted into machine code.

A *runtime error* is a program error that occurs while the program is running.

The term is often used in contrast to other types of program errors, such as syntax errors and compile-time errors. Runtime errors might include running out of memory, invalid memory address access, invalid parameter value, or buffer overflows/dereferencing a null pointer (to name a few), all of which can be discovered only by running the program as a user. Another potential runtime error can occur if there is an attempt to divide by zero. These types of errors result in a software exception. Software and hardware exceptions need to be handled properly. Consequently, structured exception handling (SEH) is a mechanism used to handle both types of exceptions. It enables the programmer to have complete control over how exceptions are handled and provides support for debugging.

Code issues and errors that occur at either compile time or run time could lead to vulnerabilities in the software. However, it's the runtime environment that we are more interested in from a security perspective because that more often is where attackers will attempt to exploit software and websites.

Input Validation

Input validation is very important for website design and application development. *Input validation*, or data validation, is a process that ensures the correct usage of data: it checks the data that is inputted by users into web forms and other similar web elements. If data is not validated correctly, it can lead to various security vulnerabilities including sensitive ***data exposure*** and the possibility of data corruption. You can validate data in many ways, from coded data checks and consistency checks to spelling and grammar checks, and so on.

Attackers can perform ***server-side execution*** by leveraging vulnerabilities such as command injection, cross-site request forgery, and SQL injection. In other words, attackers can manipulate an application by providing crafted input to an application to perform a malicious transaction, steal sensitive data, or cause a denial-of-service (DoS) condition. Similarly, attackers can perform ***client-side execution*** by exploiting vulnerabilities that can perform malicious transactions in a user's endpoint, web browser, or mobile device. All application programming interfaces (APIs) and language functions that

accept data can be potentially sent malicious input to cause the device or client application to crash or—what is worse—perform code execution.

Whatever data is being dealt with, it should be checked to make sure it is being entered correctly and won't create or take advantage of a security flaw. If validated properly, bad data and malformed data will be rejected. Input validation should be done both on the client side and, more importantly, on the server side.

If an organization has a web page with a PHP-based contact form, for example, the data entered by the visitor should be checked for errors or maliciously typed input. The following piece of PHP code is contained within a common contact form:

```
else if (!preg_match('/^[A-Za-z0-9.-]+$/', $domain))
 {
 // character not valid in domain part
 $isValid = false;
 }
```

This snippet is part of a larger piece of code that checks the entire email address a user enters into a form field. This particular code snippet checks to make sure the user is not trying to enter a backslash in the domain name portion of the email address. A backslash is not allowed in email addresses and could be detrimental if used maliciously. In the first line within the brackets, the code says **A-Za-z0-9.-**, which tells the system what characters are allowed. Uppercase and lowercase letters, numbers, periods, and dashes are allowed, but other characters such as backslashes, dollar signs, and so on are not allowed. Those other characters would be interpreted by the form's supporting PHP files as illegitimate data and would not be passed on through the system. The user would receive an error, which is a part of ***client-side validation***. But the more a PHP form is programmed to check for errors, the more it is possible to have additional security holes. Therefore, ***server-side validation*** is even more important. Any data that *is* passed on by the PHP form should be checked at the server as well. In fact, an attacker might not even be using the form in question but might be attacking the URL of the web page in some other manner. This type of attack can be checked at the server within the database software or through other means. By the way, the concept of combining client-side and server-side validation also goes for

pages that utilize JavaScript.

This is just one basic example, but as mentioned previously, input validation is the key to preventing attacks such as SQL injection and XSS. All form fields should be tested for good input validation code, both on the client side and the server side. By combining the two and checking every access attempt, you develop complete mediation of requests.

Tip

Using input validation is one way to prevent sensitive data exposure, which occurs when an application does not adequately protect personally identifiable information (PII). Such exposure can also be prevented by making sure that inputted data is never stored or transmitted in clear text; using strong encryption and securing key generation and storage; using HTTPS for authentication; and using a salt, which is random data used to strengthen hashed passwords.

Static and Dynamic Code Analysis

Static code analysis is a type of debugging that is carried out by examining the code *without* executing the program. This analysis can be done by scrutinizing code visually or with the aid of specific automated tools—static code analyzers—based on the language being used. Static code analysis can help reveal major issues with code that could even lead to disasters. Although this phase of testing is important, it should always be followed by some type of dynamic analysis—for example, fuzz testing (covered next). At this point, the program is actually executed while it is being tested. It is meant to locate minor defects in code and vulnerabilities.

Fuzz Testing

Fuzz testing (also known as fuzzing or dynamic analysis) is another smart concept. In this type of analysis, random data is inputted into a computer program in an attempt to find vulnerabilities. This is often done without knowledge of the source code of the program. The program to be tested is run, has data inputted to it, and is monitored for exceptions such as crashes.

This analysis can be done with applications and operating systems. It is commonly used to check file parsers within applications such as Microsoft Word and network parsers that are used by protocols such as DHCP.

Fuzz testing, or fuzzing, can be used to find software errors (or bugs) and security vulnerabilities in applications, operating systems, infrastructure devices, IoT devices, and other computing devices. Fuzzing involves sending random data to the unit being tested in order to find input validation issues, program failures, buffer overflows, and other flaws. Tools that are used to perform fuzzing are referred to as "fuzzers." Examples of popular fuzzers are Peach, Munity, American Fuzzy Lop, and Synopsys Defensics.

The Mutiny Fuzzing Framework is an open-source fuzzer created by Cisco. It works by replaying packet capture files (PCAPs) through a mutational fuzzer. You can obtain more information about Mutiny Fuzzing Framework at https://github.com/Cisco-Talos/mutiny-fuzzer. The Mutiny Fuzzing Framework uses Radamsa to perform mutations. Radamsa is a tool that can be used to generate test cases for fuzzers. You can download additional information about Radamsa at https://gitlab.com/akihe/radamsa.

American Fuzzy Lop (AFL) is a tool that provides features of compile-time instrumentation and genetic algorithms to automatically improve the functional coverage of fuzzing test cases. You can obtain additional information about AFL at http://lcamtuf.coredump.cx/afl/.

Peach is one of the most popular fuzzers in the industry. There is a free (open-source) version, the Peach Fuzzer Community Edition, and a commercial version. You can download the Peach Fuzzer Community Edition and obtain additional information about the commercial version at https://gitlab.com/gitlab-org/security-products/protocol-fuzzer-ce.

Tip

For additional examples of fuzzers and fuzzing, see the GitHub repository at https://h4cker.org/github.

Fuzz testing can uncover full system failures, memory leaks, and error-handling issues. Fuzzing is usually automated (a program that checks a program) and can be as simple as sending a random set of bits to be inputted

to the software. However, designing the inputs that cause the software to fail can be a tricky business, and often a myriad of variations of code must be tried to find vulnerabilities. Once the fuzz test is complete, the results are analyzed, the code is reviewed and made stronger, and vulnerabilities that were found are removed. The stronger the fuzz test, the better the chances that the program will not be susceptible to exploits.

As a final word on this type of analysis, after code is properly tested and approved, it should be reused whenever possible. This way, you can avoid reinventing the wheel and prevent common mistakes that a programmer might make that others may have already fixed. Just remember, ***code reuse*** is applicable only if the code is up to date and approved for use. Code reuse and ***dead code*** can introduce security risks if the code that is being shared is not properly tracked and patched. For instance, you may be patching vulnerabilities in one product or application that uses the original code, but not in others that reuse that code.

Programming Vulnerabilities and Attacks

The following sections address program code vulnerabilities and the attacks that exploit them. They also give specific ways to mitigate these risks during application development, hopefully preventing these threats and attacks from becoming realities.

Note

This relatively short discussion covers a topic that could fill several books. It is not a seminar on how to program, but rather a concise description of programmed attack methods and some defenses against them. The Security+ objectives do not go into great detail concerning these methods, but CompTIA does expect you to have a broad and basic understanding.

Testing for Backdoors

Applications should be analyzed for backdoors. Backdoors are used in computer programs to bypass normal authentication and other security mechanisms in place. They can be avoided by updating the operating system and applications and firmware on devices, and especially by carefully checking the code of the program. If the system is not updated, a malicious person could take all kinds of actions via the backdoor. For example, a company's software developer could install code through a backdoor that reactivates the user account after it is disabled (for whatever reason—termination, end of consulting period, and so on). This is done through the use of a logic bomb (in addition to the backdoor) and could be deadly when the software developer has access again. To reiterate, you should make sure the operating system is updated and also consider job rotation, where programmers check each other's work.

Memory/Buffer Vulnerabilities

As you learned in Chapter 3, *memory management* and buffer vulnerabilities have been used by attackers to compromise systems in different ways. Of the several types, perhaps most important is the buffer overflow. A buffer overflow occurs when a process stores data outside the memory that the developer intended. This overflow could cause erratic behavior in the application, especially if the memory already had other data in it.

Tip

Stacks and heaps are data structures that can be affected by buffer overflows. The stack is a key data structure necessary for the exchange of data between procedures. The heap contains data items whose size can be altered during execution.

In Chapter 3 you also learned what *integer overflows* and *memory leaks* are.

Another potential memory-related issue deals with pointer dereferencing—for example, the null pointer dereference. Pointer dereferencing is common in programming; when you want to access data (say, an integer) in memory, dereferencing the pointer would retrieve different data from a different section of memory (perhaps a different integer). Programs that contain a *null*

pointer dereference generate memory fault errors (memory leaks). A null pointer dereference occurs when the program dereferences a pointer that it expects to be valid but is null, which can cause the application to exit or the system to crash. From a programmatical standpoint, the main way to prevent this issue is meticulous coding. Programmers can use special memory error analysis tools to enable error detection for a null pointer deference. Once the problem is identified, the programmer can correct the code that may be causing these errors. But this concept can be used to attack systems over the network by initiating IP address to hostname resolutions—ones that the attacker hopes will fail—causing a return null. What this all means is that the network needs to be protected from attackers attempting this (and many other) programmatical and memory-based attacks via a network connection.

A programmer may make use of address space layout randomization (ASLR) to help prevent the exploitation of memory corruption vulnerabilities. It randomly arranges the different address spaces used by a program (or process). This can aid in protecting mobile devices (and other systems) from exploits caused by memory-management problems. While there are exploits to ASLR (such as side-channel attacks) that can bypass it and derandomize how the address space is arranged, many systems employ some version of ASLR.

XSS and XSRF

Two web application vulnerabilities to watch out for include *cross-site scripting (XSS)* and *cross-site request forgery (XSRF)*.

XSS holes are vulnerabilities that can be exploited with a type of code injection. Code injection is the exploitation of a computer programming bug or flaw by inserting and processing invalid information; it is used to change how the program executes data. In the case of an XSS attack, an attacker inserts malicious scripts into a web page in the hope of gaining elevated privileges and access to session cookies and other information stored by a user's web browser. This code (often JavaScript) is usually injected from a separate "attack site." It can also manifest itself as an embedded JavaScript image tag, header manipulation (as in manipulated HTTP response headers), or other HTML-embedded image object within emails (that are web-based). Programmers can defeat the XSS attack through the use of output encoding (JavaScript escaping, CSS escaping, and URL encoding), by preventing the

use of HTML tags, and by input validation: for example, checking forms and confirming that input from users does not contain hypertext. On the user side, the possibility of this attack's success can be reduced by increasing cookie security and by disabling scripts. If XSS attacks by email are a concern, the user could opt to set the email client to text only.

The XSS attack exploits the trust a user's browser has in a website. The converse of this, the XSRF attack, exploits the trust that a website has in a user's browser. In this attack (also known as a one-click attack), the user's browser is compromised and transmits unauthorized commands to the website. The chances of this attack can be reduced by requiring tokens on web pages that contain forms, using special authentication techniques (possibly encrypted), scanning .XML files (which could contain the code required for unauthorized access), and submitting cookies twice instead of once, while verifying that both cookie submissions match.

More Code Injection Examples

Other examples of code injection include SQL injection, XML injection, and Lightweight Directory Access Protocol (LDAP) injection.

Databases are just as vulnerable as web servers. The most common kind of database is the relational database, which is administered by a relational database management system (RDBMS). These systems are usually written in the Structured Query Language (SQL, pronounced "sequel"). An example of a SQL database is Microsoft's SQL Server; it can act as the back end for a program written in Visual Basic or Visual C++. Another example is MySQL, a free, open-source relational database often used in conjunction with websites that employ PHP pages.

One concern with SQL is the SQL injection attack, which occurs in databases, ASP.NET applications, and blogging software (such as WordPress) that use MySQL as a back end. In these attacks, user input in web forms is not filtered correctly and is executed improperly, with the end result of gaining access to resources or changing data. For example, the login form for a web page that uses a SQL back end (such as a WordPress login page) can be insecure, especially if the front-end application is not updated. An attacker will attempt to access the database (from a form or in a variety of other ways), query the database, find a user, and then inject code to the

password portion of the SQL code—perhaps something as simple as **X = X**. This will allow any password for the user account to be used. If the login script was written properly (and validated properly), it should deflect this injected code. But if not, or if the application being used is not updated, it could be susceptible. An attack can be defended against by constraining user input, filtering user input, and using stored procedures such as input validating forms. Used to save memory, a ***stored procedure*** is a subroutine in an RDBMS that is typically implemented as a data-validation or access-control mechanism which includes several SQL statements in one procedure that can be accessed by multiple applications.

When using relational databases such as SQL and MySQL, a programmer works with the concept of ***normalization***, which is the ability to avoid or reduce data redundancies and anomalies—a core concept within relational DBs. There are, however, other databases that work within the principle of denormalization and don't use SQL (or use code in addition to SQL). Known as NoSQL databases, they offer a different mechanism for retrieving data than their relational database counterparts. These databases are commonly found in data warehouses and virtual systems provided by cloud-based services. While they are usually resistant to SQL injection, there are NoSQL injection attacks as well. Because of the type of programming used in NoSQL, the potential impact of a NoSQL injection attack can be greater than that of a SQL injection attack. An example of a NoSQL injection attack is the JavaScript Object Notation (JSON) injection attack. But NoSQL databases are also vulnerable to brute-force attacks (cracking of passwords) and connection pollution (a combination of XSS and code injection techniques). Methods to protect against NoSQL injection are similar to the methods mentioned for SQL injection. However, because NoSQL databases are often used within cloud services, you, as security administrator, might not have much control over the level of security that is implemented. In these cases, careful scrutiny of the service-level agreement (SLA) between the company and the cloud provider is imperative.

LDAP injection is similar to SQL injection, again using a web form input box to gain access or by exploiting weak LDAP lookup configurations. The Lightweight Directory Access Protocol is used to maintain a directory of

information such as user accounts or other types of objects. The best way to protect against this (and all code injection techniques for that matter) is to incorporate strong input validation.

XML injection attacks can compromise the logic of Extensible Markup Language applications—for example, XML structures that contain the code for users. They can be used to create new users and possibly obtain administrative access. You can test for these attacks by attempting to insert XML metacharacters such as single and double quotes. They can be prevented by filtering *in* allowed characters (for example, A–Z only). This is an example of "default deny," where only what you explicitly filter in is permitted; everything else is forbidden.

One point to remember is that when attackers utilize code injecting techniques, they are adding their own code to existing code or are inserting their own code into a form. A variant of this technique is command injection, which doesn't utilize new code; instead, an attacker executes system-level commands on a vulnerable application or OS. The attacker might enter the command (and other syntax) into an HTML form field or other web-based form to gain access to a web server's password files.

Note

Though not completely related, another type of injection attack is DLL injection. In this attack, code is run within the address space of another process by forcing it to load a dynamic link library (DLL). Ultimately, this attack can influence the behavior of a program that was not originally intended.

Once again, the best way to defend against code injection/command injection techniques in general is to implement input validation during the development, testing, and maintenance phases of the SDLC.

Directory Traversal

Directory traversal, or the ../ (dot-dot-slash) attack, is a method of accessing unauthorized parent (or worse, root) directories. It is often used on web servers that have PHP files and are Linux or UNIX-based, but it can also be

perpetrated on Microsoft operating systems (in which case, it would be ..\ or the dot-dot-backslash attack). It is designed to get access to files such as ones that contain passwords. This access can be prevented by updating the operating system or by checking the code of files for vulnerabilities, otherwise known as fuzzing. For example, a PHP file on a Linux-based web server might have a vulnerable **if** or **include** statement, which when attacked properly could give the attacker access to higher directories and the passwd file.

Zero-Day Attack

A *zero-day attack* is executed on a vulnerability in software before that vulnerability is known to the creator of the software and before the developer can create a patch to fix the vulnerability. It's not a specific attack, but rather a group of attacks including viruses, Trojans, buffer overflow attacks, and so on. These attacks can cause damage even after the creator knows of the vulnerability because time may be needed to release a patch to prevent the attacks and fix damage caused by them. These attacks can be discovered by thorough analysis and fuzz testing.

Zero-day attacks can be prevented by using newer operating systems that have protection mechanisms and by updating those operating systems. They can also be prevented by using multiple layers of firewalls and by using whitelisting, which allows only known good applications to run. Collectively, these preventive methods are referred to as zero-day protection.

Table 11-2 summarizes the programming vulnerabilities/attacks covered in this section.

Table 11-2 Summary of Programming Vulnerabilities and Attacks

Vulnerability	Description
Backdoor	An attack placed by programmers, knowingly or inadvertently, to bypass normal authentication and other security mechanisms in place.
Buffer overflow	A vulnerability that occurs when a process stores data outside the memory that the developer intended.
Remote code execution (RCE)	An attack that occurs when an attacker obtains control of a target computer through some sort of vulnerability, gaining the power to execute commands on that remote computer.
Cross-site scripting (XSS)	An attack that exploits the trust a user's browser has in a website through code injection, often in web forms.
Cross-site request forgery (XSRF)	An attack that exploits the trust that a website has in a user's browser, which becomes compromised and transmits unauthorized commands to the website.
Code injection	An attack that occurs when user input in database web forms is not filtered correctly and is executed improperly. SQL injection is a very common example.
Directory traversal	A method of accessing unauthorized parent (or root) directories.
Zero day	A group of attacks executed on vulnerabilities in software before those vulnerabilities are known to the creator.

The CompTIA Security+ exam objectives don't expect you to be a programmer, but they do expect you to have a basic knowledge of programming languages and methodologies so that you can help secure applications effectively. I recommend a basic knowledge of programming languages used to build applications, such as Visual Basic, C++, C#, Java, and Python, as well as web-based programming languages such as HTML, ASP, and PHP, plus knowledge of database programming languages such as SQL. This foundational knowledge will help you not only on the exam but also when you, as security administrator, have to act as a liaison to the programming team or if you are actually involved in testing an application.

Open Web Application Security Project (OWASP)

A great resource mentioned numerous times in this book is the ***Open Web Application Security Project (OWASP)***, accessed at https://owasp.org. OWASP is a nonprofit organization that has chapters all over the world that focus on application and software security. It has numerous well-known and comprehensive projects designed to increase the awareness of secure coding, testing, and creating tools to help find and prevent security vulnerabilities.

The OWASP Testing Project is a comprehensive guide focused on web application testing. It is a compilation of many years' worth of work by OWASP members. It covers the high-level phases of web application security testing and also digs deeper into the actual testing methods used. For instance, it goes as far as providing strings for testing cross-site scripting (XSS) and SQL injection attacks. From a web application security testing perspective, it is the most detailed and comprehensive guide available. The OWASP Testing Project is available at www.owasp.org/index.php/OWASP_Testing_Project.

The OWASP Proactive Controls (www.owasp.org/index.php/OWASP_Proactive_Controls) is a collection of secure development practices and guidelines that any software developer should follow to build secure applications. These practices will help you shift security earlier into design, coding, and testing. Here are the OWASP Top 10

Proactive Controls:

1. Define Security Requirements

2. Leverage Security Frameworks and Libraries

3. Secure Database Access

4. Encode and Escape Data

5. Validate All Inputs

6. Implement Digital Identity

7. Enforce Access Controls

8. Protect Data Everywhere

9. Implement Security Logging and Monitoring

0. Handle All Errors and Exceptions

One of the most popular OWASP projects is the Top 10 Web Application Security Risks. You can find the latest Top 10 Web Application Security Risks at https://owasp.org/www-project-top-ten. The following are the OWASP Top 10 Web Application Security Risks at the time of writing:

1. Injection

2. Broken Authentication

3. Sensitive Data Exposure

4. XML External Entities (XXE)

5. Broken Access Control

6. Security Misconfiguration

7. Cross-Site Scripting (XSS)

8. Insecure Deserialization

9. Using Components with Known Vulnerabilities

0. Insufficient Logging and Monitoring

Software Diversity

Typically, organizations have different system variants that are developed or acquired simultaneously to address a wide range of business or customer requirements. This variation is referred to as ***software diversity***. Software diversity has a direct impact on all phases of software development. In some cases, this diversity could lead to an increase of complexity because this variance and diversity has to be managed in requirements analysis, design, implementation, and validation. However, in some cases, there are different benefits of having software diversity within your environment.

For example, applications or systems providing the same service but that are from different vendors or developed by different groups inside the corporation may not rely on a single operating system or architecture. In some cases, they either do not have the same vulnerability or cannot be compromised with the same exploit. Some organizations claim that the use of software diversity increases attack tolerance. Others claim that it increases complexity in the patch management process.

There is another aspect of software diversity. This is when a ***compiler*** is modified to generate variants of a ***binary*** (target application) that operates in the same way when processing benign input; however, it might operate in a different manner when given malicious input. This situation is different from the previous examples of using a diverse set of applications and technologies. This new aspect of software diversity is handled by generating variants of a program by building a binary with a diversifying compiler that can randomize the code layout, stack variables, and random allocations of heap objects at different locations in each variant.

Automation/Scripting

Automation and ***scripting*** are crucial nowadays. Companies are finding new ways to automate everything (from software development, operations, and security). Earlier in this chapter you learned about DevOps, and subsequently, the practice of secure DevOps (or DevSecOps).

A secure DevOps environment should include the following concepts:

- ***Automated courses of action*:** These courses of action ensure secure provisioning and deprovisioning of software, services, and infrastructure.
- ***Continuous monitoring*:** Monitoring ensures that applications and systems are operating correctly and securely.
- ***Continuous validation*:** This process allows you to validate applications and code in an automated fashion.
- ***Continuous integration (CI)* and *continuous delivery (CD)*:** CI is a software development practice in which programmers merge code changes in a central repository multiple times a day. CD sits on top of CI and provides a way for automating the entire software release process. When you adopt CI/CD methodologies, each change in code should trigger an automated build-and-test sequence. This automation should also provide feedback to the programmers who made the change. CI/CD has been adopted by many organizations that provide cloud services (that is, SaaS, PaaS, and so on). For instance, CD can include cloud infrastructure provisioning and deployment, which traditionally have been done manually and consist of multiple stages. The main goal of the CI/CD processes is to be fully automated, with each run fully logged and visible to the entire team.
- ***Continuous deployment*:** This concept ensures the automation of the application deployment, provisioning, and underlying network components and infrastructure.
- ***Version control*:** In modern environments, code is checked in to a version control repository (for example, GitLab or GitHub), where each change is integrated and tested with the rest of the software system. Organizations that do not have proper version control will have a hard time keeping track of bug fixes and security patches. Similarly, vendors and software providers that do not employ appropriate version control and tracking make it harder for the consumers of their technology to be able to correlate, triage, and patch security vulnerability fixes.

Elasticity and Scalability

Many organizations move to the cloud because public cloud providers have the ability to scale their services indefinitely to provide unlimited computing, storing, and networking functionality. No environment is immune to failure or ***scalability*** problems; however, large cloud providers can scale due to the gigantic size of their shared resource pools. An *elastic* infrastructure can provide a scalable solution. ***Elasticity*** is the ability of an underlying infrastructure to react to a sudden increase in demand by provisioning more resources in an automated way.

Elasticity and scalability are often achieved by deploying technologies such as load balancers and by deploying applications and resources in multiple geographical locations (data centers around the world). Other technologies, such as enabling concurrent processing (parallel processing) and automated container deployments (such as using Kubernetes) allow organizations to auto-scale. To auto-scale, systems need to be adaptive enough to perform replication of state, fail-over, and upgrades without any manual instruction, intervention, and interpretation. Additionally, modern implementations often use software infrastructure solutions including API gateways and service mesh solutions to ensure the overall system resiliency. In short, automation and orchestration are the key requirements for a reliable, scalable, and elastic IT infrastructure.

Earlier in this chapter, you learned that DevOps highlights the collaboration of software developers and IT infrastructure engineers so that the coding, testing, and releasing of software can happen more efficiently and more securely. DevOps concepts were created to shorten the software development lifecycle and provide continuous delivery through quality software. The term *elasticity* describes the ability to scale up and scale down applications. A good “elastic” environment is designed to make sure that resources are allocated appropriately and only when they are needed. With resource pooling, resources are brought together in a pooled model to provide services to many customers.

As you previously learned in this chapter, infrastructure as code allows for infrastructure configuration to be incorporated into application code, thus enabling DevOps teams to test applications in production-like environments from the beginning of the development cycle.

Chapter Review Activities

Use the features in this section to study and review the topics in this chapter.

Review Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the outer margin of the page. Table 11-3 lists a reference of these key topics and the page number on which each is found.

Table 11-3 Key Topics for Chapter 11

Key Topic Element	Description	Page Number
List	Identifying different development environments	
Paragraph	Understanding software deprovisioning	
Paragraph	Defining software integrity measurement	
Paragraph	Understanding the concept of software development lifecycle and the different SDLC components	
Paragraph	Understanding obfuscation/camouflage, memory management, and the risk associated with different third-party libraries and software development kits (SDKs)	
Paragraph	Understanding server-side vs. client-side execution.	
Paragraph	Understanding the risk associated with code reuse and dead code	
Paragraph	Defining the concept of normalization	
Section	The Open Web Application Security Project (OWASP)	
List	Understanding different elements of the secure DevOps environment	
Section	Elasticity and Scalability	

Define Key Terms

Define the following key terms from this chapter, and check your answers in the glossary:

software development environments

version control

development

testing

staging

production

quality assurance (QA)

application provisioning

deprovision

software integrity measurement

obfuscation

camouflage

memory management

third-party libraries and software development kits (SDKs)

data exposure

server-side execution

client-side execution

client-side validation

server-side validation

code reuse

dead code

stored procedure

normalization

Open Web Application Security Project (OWASP)

software diversity

compiler

binary

automation

scripting

continuous delivery

continuous integration

automated courses of action

continuous monitoring

continuous validation

continuous integration (CI)

continuous delivery (CD)

continuous deployment

version control

scalability

elasticity

Review Questions

Answer the following review questions. Check your answers with the answer key in Appendix A.

1. What type of debugging is carried out by examining the code *without* executing the program? It can be done by scrutinizing code visually, or with the aid of specific automated tools—static code analyzers—based on the language being used.
2. What is the process of measuring your source code's quality when it is passed on to the quality assessment (QA)?
3. What is the software development environment where you create your code on your computer or in the cloud?
4. What is a development environment that allows you to test your code or application but is as similar to the production environment as it can be? This environment allows you to ensure that each component of your application still does its job with everything else going on around it.
5. What is a software development and project management process where a project is managed by breaking it up into several stages and involving constant collaboration with stakeholders and continuous improvement and iteration at every stage?

Chapter 12. Summarizing Authentication and Authorization Design Concepts [This content is currently in development.]

This content is currently in development.

Chapter 13. Implementing Cybersecurity Resilience [This content is currently in development.]

This content is currently in development.

Chapter 14. Understanding the Security Implications of Embedded and Specialized Systems [This content is currently in development.]

This content is currently in development.

Chapter 15. Understanding the Importance of Physical Security Controls [This content is currently in development.]

This content is currently in development.

Chapter 16. Summarizing the Basics of Cryptographic Concepts [This content is currently in development.]

This content is currently in development.

Part III: Implementation

Chapter 17. Implementing Secure Protocols [This content is currently in development.]

This content is currently in development.

Chapter 18. Implementing Host or Application Security Solutions [This content is currently in development.]

This content is currently in development.

Chapter 19. Implementing Secure Network Designs [This content is currently in development.]

This content is currently in development.

Chapter 20. Installing and Configuring Wireless Security Settings [This content is currently in development.]

This content is currently in development.

Chapter 21. Implementing Secure Mobile Solutions [This content is currently in development.]

This content is currently in development.

Chapter 22. Applying Cybersecurity Solutions to the Cloud [This content is currently in development.]

This content is currently in development.

Chapter 23. Implementing Identity and Account Management Controls [This content is currently in development.]

This content is currently in development.

Chapter 24. Implementing Authentication and Authorization Solutions [This content is currently in development.]

This content is currently in development.

Chapter 25. Implementing Public Key Infrastructure [This content is currently in development.]

This content is currently in development.

Part IV: Operations and Incident Response

Chapter 26. Using the Appropriate Tool to Assess Organizational Security [This content is currently in development.]

This content is currently in development.

Chapter 27. Summarizing the Importance of Policies, processes, and procedures for incident response [This content is currently in development.]

This content is currently in development.

Chapter 28. Using Appropriate Data Sources to Support an Investigation [This content is currently in development.]

This content is currently in development.

Chapter 29. Applying Mitigation Techniques or Controls to Secure an Environment [This content is currently in development.]

This content is currently in development.

Chapter 30. Understanding the Key Aspects of Digital Forensics [This content is currently in development.]

This content is currently in development.

Part V: Governance, Risk, and Compliance

Chapter 31. Comparing and Contrasting Various Types of Controls [This content is currently in development.]

This content is currently in development.

Chapter 32. Understanding the Importance of Applicable Regulations, Standards, or Frameworks That Impact Organizational Security Posture [This content is currently in development.]

This content is currently in development.

Chapter 33. Understanding the Importance of Policies to Organizational Security [This content is currently in development.]

This content is currently in development.

Chapter 34. Summarizing Risk Management Processes and Concepts [This content is currently in development.]

This content is currently in development.

Chapter 35. Understanding Privacy and Sensitive Data Concepts in Relation to Security [This content is currently in development.]

This content is currently in development.

Part VI: Final Preparation

Chapter 36. Final Preparation [This content is currently in development.]

This content is currently in development.

Glossary of Key Terms [This content is currently in development.]

This content is currently in development.

Appendix A: Answers to the “Do I Know This Already?” Quizzes and Review Questions [This content is currently in development.]

This content is currently in development.

Appendix B: CompTIA Security+ SY0-601 Exam Updates [This content is currently in development.]

This content is currently in development.

Appendix C: (Website only) Study Planner [This content is currently in development.]

This content is currently in development.