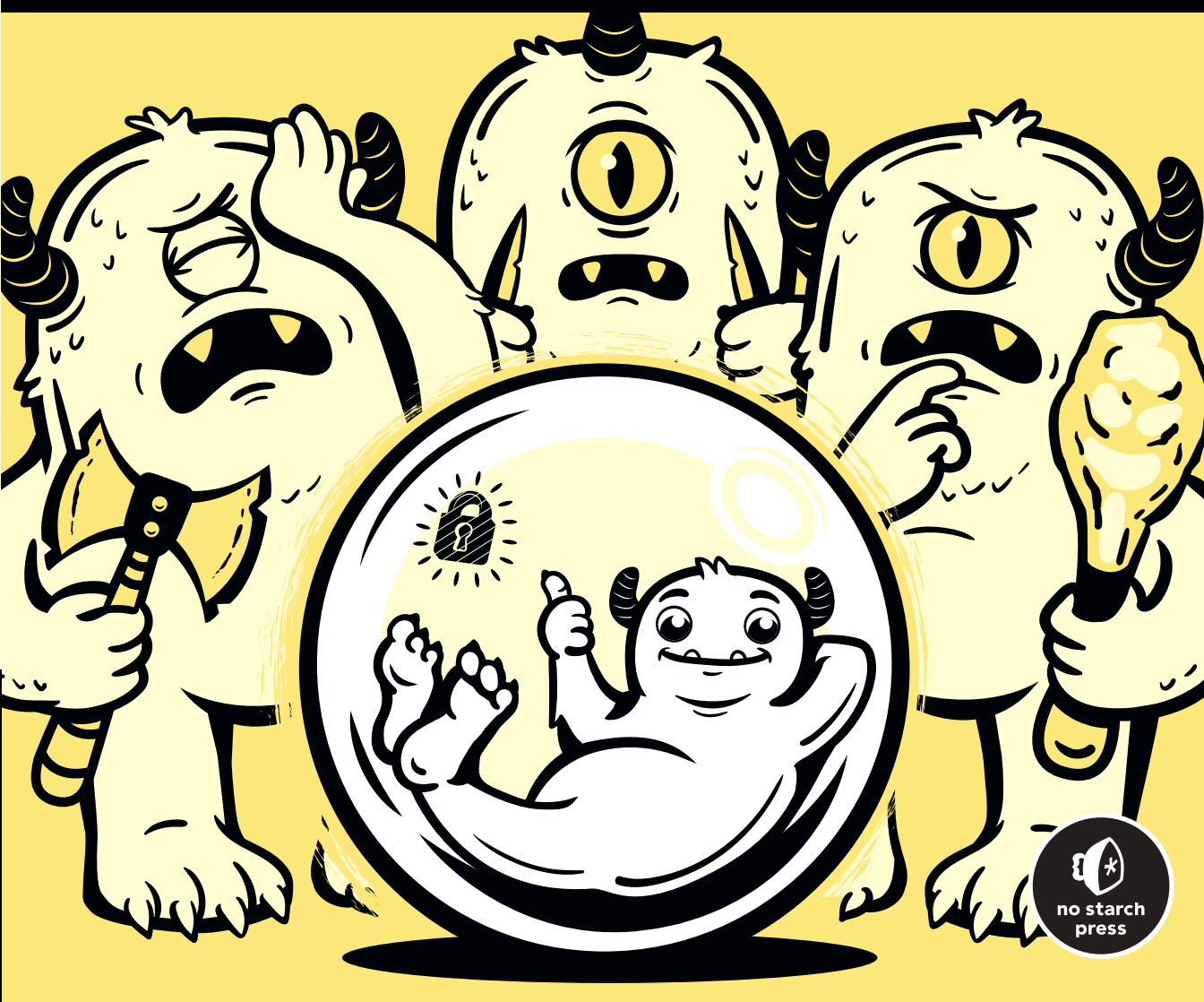
HOW CYBERSECURITY REALLY WORKS

A HANDS-ON GUIDE FOR TOTAL BEGINNERS

SAM GRUBB



no starch press

HOW CYBERSECURITY REALLY WORKS

HOW CYBERSECURITY REALLY WORKS

A Hands-on Guide for Total Beginners

by Sam Grubb

no starch press

San Francisco

ISBN-13: 978-1-7185-0128-7 (print)

ISBN-13: 978-1-7185-0129-4 (ebook)

Publisher: William Pollock
Executive Editor: Barbara Yien
Production Manager: Rachel Monaghan
Production Editor: Dapinder Dosanjh
Developmental Editor: Frances Saux
Cover Illustrator: Gina Redman
Interior Design: Octopod Studios
Technical Reviewer: Cliff Janzen
Copyeditor: Anne Marie Walker
Compositor: Craig Woods, Happenstance Type-O-Rama
Proofreader: Rachel Head

For information on book distributors or translations, please contact No Starch Press, Inc. directly:
No Starch Press, Inc.
245 8th Street, San Francisco, CA 94103
phone: 1.415.863.9900; info@nostarch.com
www.nostarch.com

Library of Congress Cataloging-in-Publication Data
Names: Grubb, Sam (Cyber security consultant), author.
Title: How cybersecurity really works : a hands-on guide for total
beginners / Sam Grubb.
Description: San Francisco : No Starch Press, 2021. | Includes index.
Identifiers: LCCN 2021004423 (print) | LCCN 2021004424 (ebook) | ISBN
9781718501287 (paperback) | ISBN 9781718501294 (ebook)
Subjects: LCSH: Computer security. | Computer crimes--Prevention. |
Computer networks--Security measures.
Classification: LCC QA76.9.A25 G78 2021 (print) | LCC QA76.9.A25 (ebook)
| DDC 005.8--dc23
LC record available at https://lccn.loc.gov/2021004423
LC ebook record available at https://lccn.loc.gov/2021004424

To Shannon and Elliott, whose love and
constant support gives me the confidence
to do more than I ever imagined.

About the Author

Sam Grubb is a cybersecurity consultant and education advocate, as well as a former teacher, librarian, sandwich artist, and military helmet researcher. He currently works with companies and healthcare providers to ensure they are meeting both security and compliance needs. He believes nothing is too difficult to learn; you just need the right teacher. He enjoys reading, writing, and staying as far away from arithmetic as possible. He lives in Arkansas with his wife, son, two cats, and two dogs.

About the Technical Reviewer

Since the early days of Commodore PET and VIC-20, technology has been a constant companion to Cliff Janzen—and sometimes an obsession! Cliff is grateful to have the opportunity to work with and learn from some of the best people in the industry, including Sam and the fine people at No Starch Press. Cliff spends a majority of his workday managing and mentoring a great team of security professionals, striving to stay technically relevant by tackling everything from security policy reviews and penetration testing to incident response. He feels lucky to have a career that is also his favorite hobby and a wife who supports him.

BRIEF CONTENTS

CONTENTS IN DETAIL

PREFACE

Look at any major news source and you're bound to find a story or two about recent cyberattacks. Whether it's a new scam spreading across the internet, hospitals or other organizations targeted with ransomware, or even elections disrupted by other countries, adversaries try to circumvent cybersecurity in many ways. You might find these attacks of interest but think they don't have much to do with you. However, as cyberattacks become increasingly common, attackers no longer focus exclusively on big corporations. They've begun to target everyday individuals. As a result, you can no longer afford to just read about cybersecurity; it's a daily skill you need to learn and practice.

If you've struggled and searched to learn about cybersecurity without first cultivating a deep technical background, look no further: this book is perfect for those who have no background in security, or even computers for that matter.

I created this book to fill a gap. Few resources exist for those who want to understand more than just the basics of computer engineering or administration but aren't trying to become full-fledged cybersecurity professionals. It's designed to cover a wide range of topics across the core cybersecurity concepts. Cybersecurity is a vast field with lots of deep valleys that you can easily get lost in. Think of this book as a helicopter tour; you'll fly over those valleys to get an idea of where you might explore next.

To provide this overview, this book focuses on how black hats operate and the sorts of attacks that exist. At its core, cybersecurity is about defending

against threats, both physical and logical, to technical assets. By focusing on what black hats attempt to do, we'll link threats to the vulnerabilities that cause them and controls that protect against them.

A Note on the Book's Exercises

The only way to learn cybersecurity concepts is to practice them. To that end, every chapter ends with an exercise that helps you apply the concepts you just learned. These exercises are designed to be completed at home and provide some insight into what you can do to make sure your systems are secure every day. They focus on the core concepts while providing practical knowledge you can use when implementing cybersecurity.

The exercises in this book assume you're using the Windows or macOS operating system, because of their widespread use by people and organizations worldwide. To follow along, you'll need at least a Windows 10 or macOS X system.

Many cybersecurity professionals and tools use Linux-based operating systems instead. Although this book doesn't cover Linux, with a little research you can easily translate many of the concepts explained in the exercises to a Linux system. If you want to pursue cybersecurity further after reading this book, I encourage you to learn about Linux by using resources like *Linux Basics for Hackers* by OccupyTheWeb (No Starch Press, 2019).

Who This Book Is For

This book is for anyone who's interested in cybersecurity but isn't entirely sure what cybersecurity means. That includes people without technical backgrounds, although if you're just beginning your technical career or are a new computer science student interested in cybersecurity, this book is definitely a great place to start. The intended audience also includes business leaders, account managers, sales and marketing professionals, or any hobbyist who might want to understand why cybersecurity is so important and what it encompasses.

Readers of any age will benefit from reading this book. Although knowledge of some basic concepts about how computers or networking works is helpful, it's not required to understand the topics in this book. Most of all, this book is for anyone who has ever been curious about how hacking or cybersecurity works in the real world, far beyond what you see in movies or on TV.

What's in the Book?

The following breakdown of each chapter gives you an idea of the topics we'll explore:

Chapter 1: An Introduction to Cybersecurity This chapter explains what cybersecurity is and isn't, the different roles and responsibilities of

cybersecurity professionals, and various types of adversaries who attack computer systems. In the chapter's exercise, you'll set up threat feeds to learn more about attackers' activities globally.

Chapter 2: Attack Targets on the Internet This chapter covers how adversaries find you on the internet and includes a primer on how the internet works. You'll learn what attackers do to find your computer or network using basic information, what attack methodology they often follow, and what you can do to hide from them online. It ends with an exercise that shows you how to use command line tools to discover information about your computer and network.

Chapter 3: Phishing Tactics This chapter focuses on social engineering attacks that exploit human behavior. It covers different types of phishing, how black hats attempt to trick you into thinking they're someone else, and how you can recognize these types of attacks. As an exercise, you'll analyze emails to determine whether they're black hat tricks.

Chapter 4: Malware Infections This chapter describes malware and other kinds of nasty software that black hats use to gain access to your system. I'll describe different types of malware and what you can do to prevent malware infections. In this chapter's exercise, you'll safely analyze files to see whether they contain malware.

Chapter 5: Password Thefts and Other Account Access Tricks This chapter covers authentication: in other words, how you log in to a computer or accounts online. We'll explore the types of authentication and the kinds of attacks adversaries use to break it. Then we'll discuss what you can do to make sure your account security remains strong. The chapter's exercise teaches you how to set up secure authentication for Windows and macOS systems.

Chapter 6: Network Tapping This chapter explores how adversaries attack your network to find data or stop you from using the internet. It explains how wired networks work, how attackers use that knowledge to their advantage, and what you can do to stop these attacks. In the exercise, you'll set up the default firewall installed on Windows and macOS devices.

Chapter 7: Attacks in the Cloud This chapter discusses what cloud computing means. It then looks at ways that adversaries attack the cloud, including ways that they attack web applications. It also provides methods you can use to secure your cloud accounts against attacks. As an exercise, you'll practice performing a SQL injection attack to better understand how attackers use these.

Chapter 8: Wireless Network Pirating This chapter covers everything wireless-related: what wireless is, how it works, how adversaries attack the wireless internet, and the best ways to stay safe. It ends with an exercise on how to secure a wireless router from attacks.

Chapter 9: Encryption Cracking This chapter explains encryption, how we use it, and what attackers do to break it. We'll cover different

types of encryption and attacks used to break each. We'll also discuss how you can ensure your systems use encryption correctly. In the chapter's exercise, you'll learn how to encrypt and hash files.

Chapter 10: How to Defeat Black Hats This chapter summarizes the concepts discussed throughout the book in the context of risk management practices. You'll learn how to manage all the controls and defense measures covered in the book to make sure you have a comprehensive security program. As an exercise, you'll create a risk management plan to guarantee you have the proper security in place to try to prevent attacks.

By the end of this book, you'll have a solid idea of what cybersecurity includes, what the core concepts are, how specific attacks work (and what controls you can use to defend your system against them), and how you can implement cybersecurity in practice. You'll be ready to move on to more advanced topics based on your interests, whether they involve learning how to implement an Active Directory server, create your own encryption cipher, manage vulnerabilities, or run penetration tests. The best part is that you'll understand how cybersecurity can affect your everyday life and what you can do to secure your devices against increasingly common black hat attacks.

ACKNOWLEDGMENTS

I would like to first thank my family. Without the support and faith of my wife, Shannon, I would never have been able to finish this book, let alone put a single word down. I'd also like to thank my parents and sister, who helped me develop the original idea for this book. Without all of their support, it would still be nothing but a joke about cats hacking the internet.

I would like to thank Bill Pollock, Barbara Yien, and the team at No Starch Press. They took a chance on me and made my dream of writing a book on cybersecurity a reality. For that, I will be eternally grateful. I'd also like to acknowledge Frances Saux and Cliff Janzen. Without their editing and insight, this book would just be a pile of words. Finally, I'd like to acknowledge my fellow consultants at Edafio, whose constant mentoring and teaching make me better at security every day.

1

AN INTRODUCTION TO CYBERSECURITY

Cybersecurity is a vast and diverse field. Whether you're setting up a firewall or creating a password policy, your actions impact all levels of an organization, from its technicians and help desk to the CEO. Cybersecurity also affects every piece of technology in an organization: mobile phones, servers, and even devices like industrial control systems. A field this extensive and deep can be a little intimidating when you first enter it. This is especially true if you're trying to learn about cybersecurity without entering the field. For example, you might be an IT department head who wants to learn more so you can better protect your organization.

This chapter starts slow: we'll talk about what cybersecurity is and isn't, as well as the difference between white hat and black hat hackers.

What Is Cybersecurity?

At its core, cybersecurity has one driving purpose: to identify cyber threats in an organization, calculate the risk related to those threats, and handle those threats appropriately. Not every threat that a company experiences is an issue that cybersecurity deals with directly (for example, pandemics or physical damage to a building caused by a tornado or flood). In general, cybersecurity uses the *CIA triad* model to determine which threats are under its purview.

The CIA triad consists of three categories of security: confidentiality, integrity, and availability. *Confidentiality* involves how assets and data are exposed to people or processes, and ensures that only the people who are supposed to access a resource can access it. *Integrity* ensures that assets and data aren't changed without proper authorization. This not only includes items like entries in a database server, but also adding a user to a network, for example. *Availability* ensures that data or assets are accessible when needed. For work to continue, you must be able to access data when necessary.

Figure 1-1 shows the elements of the CIA triad positioned in a triangle to demonstrate how you might need to balance each of them to maintain the functionality of the others. For example, if you focus too much on confidentiality, you risk significantly locking down your assets so no one else can use that data for their job, creating an availability issue. Similarly, by placing too much emphasis on integrity, you lose confidentiality, because you must be able to read data to ensure that nothing has changed. By balancing the three triad components, you can achieve equilibrium between the core elements that encompass what cybersecurity does on a regular basis.

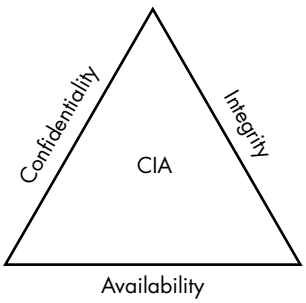


Figure 1-1: The CIA triad

Some experts debate the merits of adding elements to the traditional triad to contend with new technologies or priorities within cybersecurity. One element often added is *non-repudiation*, which is the idea that when a person or entity does something, there must be specific evidence tying them to that action so it's impossible for them to deny they did it.

Cybersecurity and Privacy

In recent years, there has been an emphasis on the relationship between cybersecurity and privacy. In this situation, privacy means the rights and

abilities of a person to control how information about them is stored, shared, and used. Although the topic of privacy extends beyond cybersecurity, cybersecurity still plays a huge role in ensuring that an individual's data is secured against malicious use. Cybersecurity is also responsible for many of the controls that allow a company to audit its data use, ensuring that it follows any necessary rules or regulations. Going forward, the protection of a user's privacy will likely become an increasingly integral part of the cybersecurity field.

What Cybersecurity Isn't

In a field as large as cybersecurity, you're bound to encounter a few distorted ideas about its scope. To mitigate these misconceptions, it's best to discuss what cybersecurity isn't. Doing so will help define the field and what it actually means to *do* cybersecurity.

First, cybersecurity isn't synonymous with hacking. The media would have you believe that all cybersecurity professionals do is clack away at a keyboard, trying to break into a system. Although *penetration testing*—the act of attempting to break into a system you're authorized to attack, such as your own or a client's, to discover vulnerabilities from an attacker's perspective—is a part of cybersecurity, it's but one section of the field. A *vulnerability* is a flaw in a system, including how it's set up or how people use it. For example, having an error in a system's code can cause a vulnerability. Attackers create *exploits* to take advantage of vulnerabilities. But just because you don't know how to execute an exploit using a flaw in a computer's memory doesn't mean you can't be an expert in setting up and maintaining firewalls. This means that you don't need to understand how every hacking tool works or exactly what the latest exploit does to contribute to the cybersecurity industry.

Second, cybersecurity isn't switch flipping. Some people use the term *switch flipping* to describe what they think system engineers or other IT professionals do: they just flip switches or configure systems without understanding the underlying processes that make a system work. It's true that configuring a system to be secure is vitally important to cybersecurity. But securing a system can't necessarily be done by following a checklist. It requires looking at the entire system, noting how every component interacts not only with the other components, but also with other systems to fully understand how to secure a system. In addition, professionals need deliberation and critical thinking skills to know how to secure a system in situations where it's impossible to apply best practices.

Third, cybersecurity doesn't only require technical skills. Just as important as technical knowledge is the ability to translate that information into tips and resources that everyone can understand when professionals give presentations or write reports. Cybersecurity professionals work with every department in an organization, which means their interpersonal communication skills are essential. The only way your organization will become more secure is if everyone understands their role in maintaining security, which means you must communicate that role effectively.

Black Hats vs. White Hats

When you think of the term *hacker*, you probably think of someone doing something malicious to or with a computer, such as destroying files or unlocking electronic locks on doors so robbers can break in. The reason you think this way is that the media generally uses the word *hacker* to describe computer criminals. But not all hackers are hoodie-clad teenagers in basements banging on a keyboard while listening to death metal. In fact, people from all different backgrounds and regions participate in computer crime. The term *hacker* is also used to describe good cybersecurity experts: the label applies to anyone who asks questions and breaks systems, whether they're computers or physical devices, to learn more about them, not necessarily just to commit crimes. Many specific expressions, such as *bad actor, attacker,* and *state actor,* single out cybercriminals. But in this book, I'll call them *black hats* (as well as *attackers* or *adversaries*).

As just mentioned, attackers come from different backgrounds and places, but they all share the same intent: to use their technical knowledge to commit a crime. These crimes often revolve around financial gain of some sort, either directly by stealing money or demanding ransom payments, or indirectly by stealing important information, such as social security numbers to sell at a later time. It's important to note that not every adversary is pursuing money. They could be seeking specific information or trying to disrupt a service. There are many arguments about what constitutes a crime when it comes to malicious computer use. For the purposes of this book, I consider any violation of the current United States Computer Fraud and Abuse Act to fit the definition of cybercrime.

On the other side of the spectrum are the white hats. *White hats* are cybersecurity experts who apply their technical knowledge to making systems more secure. They not only include people who work for a company's security department, but also independent professionals who conduct security research, such as analyzing malware or discovering *zero-day* vulnerabilities (brand-new, never-before-seen vulnerabilities in a system or software). These people work tirelessly to try to stay one step ahead of black hats.

In a gray area in the middle are *gray hats*. The activities of a gray hat aren't necessarily malicious, but they're not honorable either. For example, attacking a system without permission to find vulnerabilities that you then disclose to the system's owner is a gray area, because typically white hats don't perform any attacks without permission. Which side a gray hat falls on depends on a person's perspective. If someone uses their skills to get past a government filter on the internet, they might look like an attacker to the government but a white hat to everyone trying to exercise freedom of speech.

Types of Black Hats

Although a wide variety of people fit the role of a black hat, you can still group them into categories. These categories are not meant to be exhaustive but should give you a general idea of the motivations behind black hat activity.

Script Kiddies

Script kiddies are adversaries who have no inherent skill and follow instructions found on the internet to execute their attacks. They generally find prewritten scripts (hence the name *script kiddie*) built to run a specific type of attack. They then enter their target information and fire off the script. Traditionally, script kiddies pose a low threat to most organizations. The attacks they use aren't usually sophisticated and often rely on outdated or easily recognized vectors of attack. But script kiddies shouldn't be taken lightly. Just because they don't have the skills of more elite black hats doesn't mean they can't do damage given the right set of tools.

Organized Criminals

A growing sector of organized crime is turning to black hat activities as government policing cuts off their other sources of revenue. Organized crime is highly effective at recruiting people with expert skills. As a result, these attackers use the latest vulnerabilities, create their own malware, and do extensive research to obtain large financial payoffs for their work. This makes them significant threats. Eastern Europe and Russia are particular hotbeds for this type of activity.

Hacktivists

A *hacktivist* is a person or group who uses hacking skills for a political purpose. They usually try to deface or disrupt services rather than stealing data or money. For example, a hacktivist group might take possession of the Twitter account of a company they disagree with, using the account to write terrible messages to smear the company's reputation or promote their own agenda. One of the most legendary hacktivist groups is Anonymous, which generally targets governments or other organizations it believes are authoritative in nature. It has taken down websites and released leaked documents, among a number of other activities (although it's hard to know exactly what the group has accomplished, because anyone can claim to be a member). Hacktivists can pose a significant threat to organizations and are generally more skilled than script kiddies.

State Actor

A *state actor* is a black hat who works for a government. To many, these agents operate in the gray area, because the legitimacy of their actions might seem to vary depending on which government they happen to work for. Nevertheless, state actors use the same techniques as other attackers, and their attacks can cause significant damage. State actors are typically interested in either stealing proprietary information to help their nation or disrupting services to hurt a foreign nation. China, North Korea, Iran, and Russia have robust programs connected to several major black hat campaigns, including breaches into Sony to steal sensitive internal documents

and disruptions to elections worldwide. State actors pose some of the highest risks because they're well funded and they operate with the latest technology and training.

Advanced Persistent Threats

A more recent term, an *Advanced Persistent Threat (APT)* describes an attack that remains hidden for an extended period, slowly digging deeper into its target system until it meets its goals. Originally, state actors were the only types of adversaries with the resources and expertise to perform this type of attack. But in recent years several non-government groups have been able to execute similar operations. APTs are extremely dangerous, because it's difficult to identify where they are in your organization, what they might have access to, or who they've compromised. APTs run the gamut of motivation from targeted data theft to straight ransom.

Types of White Hats

Just like black hats, white hats fill a diverse variety of roles needed for a successful cybersecurity program. Cybersecurity isn't a monolith; it covers a multitude of fields and areas of expertise, and it's extremely difficult for one person to handle it alone. Organizations that cannot afford a dedicated security team should consider seeking outside help to supplement their own internal IT staff and provide advice where required.

The following sections explain various white hat positions along with a brief description of the typical tasks of each position. This list is by no means exhaustive, nor should it be considered standard, because some organizations might have different needs or differing ideas about where a position fits in their internal structure. That said, this list should provide you with a good idea of the types of positions that exist and the skills a person needs to fill specific roles. Also, note that I don't mention any educational degrees. The reason is that most roles in cybersecurity don't require any specific degree; instead, they rely heavily on knowledge and experience (both of which can be accumulated elsewhere). I've encountered experts with advanced cybersecurity degrees and others who had master's degrees in military history. Even so, it might take longer to gather the necessary knowledge and experience without a degree.

Cybersecurity/Security Operations Center Analysts

A *cybersecurity analyst* is an entry-level role tasked with maintaining and monitoring alerts that come in from various cybersecurity tools or devices. Their primary job is to find anything that looks suspicious and send it up the chain for further analysis if necessary. Often, these roles are tied into a *Security Operations Center (SOC)*, a facility where systems aggregate and monitor alerts from across an organization.

Analysts are the first responders for many security incidents, because they're the ones getting the alerts or directly contacting end users. These jobs typically require a strong IT background: additional security experience

is beneficial, but it's not always required. To be successful in this position, a person needs a solid understanding of networking or system administration, attention to detail, patience, and problem solving and task management skills.

Cybersecurity Consultants

Cybersecurity consultants provide a wide range of services and require an extensive background in security. Essentially, they're tasked with providing security expertise to an organization for whatever task or problem the organization is currently dealing with. This includes issues such as policy creation, system security controls, incident response, training and awareness, and general security advice. Consultants require a deep understanding of the overarching principles of security and typically have a base knowledge of most operating systems, software, or specific hardware devices. Critical thinking, problem solving, excellent verbal and written skills, and task management skills are essential for this position.

Cybersecurity Architects

We typically think of an architect as someone who designs buildings. A *cybersecurity architect* has a similar job, but instead of buildings, they design security. They're tasked with creating security controls for environments rather than implementing or managing existing controls. This means they must have a complete understanding of how security controls work and of the environment they're working with, as well as how that environment and the controls within it interact during normal workflow. For example, a network security architect would design the security controls that protect a particular network environment, taking into account the security devices needed, how information flows across the network, and any necessary network security controls on individual systems.

If you think this sounds like a sizable and complex job, you're right. Cybersecurity architects must have a vast amount of experience in their particular area of expertise, such as networking or databases, in addition to a robust security background. Understanding what controls a workflow needs and how those controls might have adverse interactions with other parts of an environment requires high-level critical thinking and problem solving skills. Architects must also work with diverse teams that span every aspect of IT, so they must hone their written and verbal communication skills. Additionally, architects are often working against a production timeline, which means they need to be efficient but diligent in their work.

Chief Information Security Officers

Organizations generally have a group of people tasked with running all operations. These people hold titles such as chief executive officer (CEO), chief financial officer (CFO), or chief information officer (CIO). In the security sector, the comparable position is the *chief information security officer (CISO)*. The CISO oversees all security operations within an organization: they make

broad decisions about how the organization should manage its security and what projects or resources the company needs to ensure it maintains an adequate level of security for the threats it faces.

The CISO requires an extensive understanding of security, but what sets them apart from most security professionals is their other skills. To be a CISO, you need excellent project management skills and budgeting experience. You also need to be able to communicate with your team and other executive officers to explain the organization's goals and mission, as well as how security relates to them. CISOs spend a good amount of their time as managers, whether with respect to personnel, budgets, or risk. Risk management requires you to identify a threat, the impact of that threat on the organization, the likelihood of that threat being realized, and what you can do to mitigate it (Chapter 10 covers risk management in depth). As the head of security for your organization, strong leadership skills are also a must.

Even small organizations need a CISO. Having a person in this role, whether it's a full-time job or part of other duties they carry out, is integral to building and maintaining security. Smaller organizations might consider finding a consultant to provide CISO-level guidance on a part-time basis.

CYBERSECURITY SPECIALTY AREAS

Although many careers in cybersecurity require a diverse knowledge of information technology, there are also many specialty areas that focus solely on one type of system or environment. For example, if you're a network administrator with a background in Cisco hardware, you might focus primarily on network security. If you're interested in malware, you might focus on malware analysis. Keep in mind that focusing on a specific area doesn't mean you can ignore security concepts related to other areas. Having a broad understanding of security in general will help reinforce the skills you learned for your particular area of expertise.

Incident Responders

An *incident* is anything bad that happens to an organization: for example, an account is compromised, data is lost or destroyed, or malware has infected a system. *Incident responders* are the people who react when an incident happens. Their main job is to run an initial investigation, preserve information and evidence, contain the incident from spreading, and restore affected systems as quickly as possible. You can compare an incident responder to a paramedic. Paramedics stabilize an injured person and determine how they were hurt so the doctor can fully treat them. Incident responders are somewhat similar: they don't perform the full investigation into what happened. That is left up to forensics experts, which we'll look at shortly.

Instead, incident responders stabilize the systems where the incident occurred to ensure the attack doesn't spread across the entire environment. For example, they might take a system off the network to stop the spread

of malware. Incident responders then gather and preserve evidence of the incident. This means checking logs, copies of systems, backups, and whatever other information they can find. Once they've gathered all the data and have the incident contained, they work to restore the environment. This might mean wiping a system to remove any possible trace of malware, for example.

Incident responders must work quickly but methodically. They require a cool head under pressure. They must be critical thinkers capable of reasoning through every action to ensure they don't make the incident worse or destroy evidence. Incident responders usually have a strong security background but often require additional training in specific incident response techniques. Responders typically work in a large team. Often, they're called on to provide specific system expertise for that team; for example, they might have an in-depth understanding of Linux operating systems.

Vulnerability Managers and Threat Hunters

Whereas incident response is about reacting to a harmful occurrence, vulnerability management attempts to prevent adverse events before they happen. *Vulnerability managers* look for security flaws in systems and try to correct them. This is a constant process, because systems continuously change and thus develop new vulnerabilities. A vulnerability manager needs patience and diligence, leaving no stone unturned to ensure they leave no vulnerabilities undiscovered.

Threat hunters have similar jobs, but they operate on a deeper scale, attempting to correlate events from across an organization to detect possible threats. They often look for advanced black hat activity, such as that carried out by an APT and not normally identified by typical alerts. Threat hunters require deep security knowledge, an eye for details, and excellent critical thinking skills. They also need good verbal and written communication skills to inform everyone in the organization about the threats they're detecting.

Computer Forensic Analysts

After an incident takes place and incident responders have completed their job, the forensic analysis of the incident begins. *Computer forensic analysis* is the process of retrieving and analyzing evidence related to an incident.

Computer forensic analysts could be part of the incident response team but are often a separate group that takes over the investigation after the threat has been contained. These analysts do a deep and detailed investigation of the evidence gathered. Not only do they look at items like logs, but they also examine the processes that were running on a system, what was loaded into memory during the incident, and even individual software code. This requires an extremely technical background with an in-depth knowledge of the inner workings of computer code. Computer forensic analysts use a variety of specialty tools that require training and

practice to use effectively. They must have an intense attention to detail, as well as good communications skills to relay their findings in language accessible to nontechnical people.

Penetration Testers

The quintessential role for most people with cybersecurity expertise is *penetration tester*. They try to break into a system as if they were black hats to discover the system's flaws and vulnerabilities. Penetration testing is actually a minor field that requires a great deal of training to be successful.

Penetration testers must have robust technical skills, because they must understand security concepts and the types of techniques attackers use. This requires constant training and practice. Penetration testers rely on a variety of tools to attack systems, each of which comes with its own set of expertise. It's also essential that they maintain meticulous documentation to provide evidence of their actions to the client: ultimately, breaking into a system doesn't matter if you can't explain how you did it.

Exercise: Learning More About Cybersecurity and Threats

To better understand cybersecurity, it helps to get involved in the community. The best way to do this is to sign up for newsletters and alerts. The following sections provide a list of some of the best feeds available to get you started. As you look through these resources, try to answer these questions: What types of threats are most common? How do various sources categorize these threats? What common advice can you find across different resources to prevent attacks? What sorts of search terms might you use to find more resources?

Government resources

National Institute of Standards and Technology (NIST) Computer Security Resource Center at *https://csrc.nist.gov/*: a great place to find articles and other information on how to secure your systems at home or work.

Cybersecurity and Infrastructure Security Agency (CISA) at *https://www.cisa.gov/*: the government agency charged with providing guidance on cybersecurity and infrastructure security. The site contains lots of resources and bulletins on security practices and threats.

National Institute for Cybersecurity Education (NICE) at *https://www.nist.gov/itl/applied-cybersecurity/nice/*: part of NIST, this group provides educational resources related to cybersecurity, including challenges and training courses for middle and high school students.

Threat feeds

Multi-State Information Sharing and Analysis Center (MS-ISAC) at *https://www.cisecurity.org/ms-isac/*: this site provides alerts on critical vulnerabilities and other information related to cybersecurity.

InfraGard at *https://www.infragard.org/*: this program provides national and state organizations with threat intelligence as well as other services, including training.

SANS Internet Storm Center at *https://isc.sans.edu/*: this site provides updates on security vulnerabilities and blog posts on various security topics.

Cybersecurity blogs

Krebs on Security at *https://krebsonsecurity.com/*: written by security expert Brian Krebs, this site provides lots of informative articles on current threats and other cybersecurity trends.

Threatpost at *https://threatpost.com/*: this site provides articles on the latest vulnerabilities and threats being exploited.

FireEye Blogs at *https://www.fireeye.com/blog.html*: this site contains information on threats, stories from the industry, and other valuable cybersecurity articles.

Cybersecurity podcasts

Security Now at *https://twit.tv/shows/security-now/*: hosted by Leo Laporte and Steve Gibson, this cast delves deeply into the headlines of the week related to cybersecurity. It's a great resource to use to keep up with the latest vulnerabilities, exploits, and threats.

Darknet Diaries at *https://darknetdiaries.com/*: created by Jack Rhysider, this cast investigates real-life stories of hackers and other security events over the years.

Conclusion

Cybersecurity can be an intimidating field to enter. However, faced with a wide variety of attackers and threats, organizations need cybersecurity professionals more than ever if they want to preserve their security. This chapter introduced you to what cybersecurity is and the threats that exist. The rest of the book will guide you through the cybersecurity field and the threats you might encounter, whether you're a manager, a long-time IT person switching to a new field, or someone just entering the professional world.

2

ATTACK TARGETS ON THE INTERNET

You know what types of black hats exist, but a question still remains: how do they find you? Most people don't expect to be targeted by an attacker. You might wonder what you have that a black hat wants.

You'd be surprised at what an attacker finds valuable. It's true that many steal credit card and social security numbers, but others look for more than just personal data. Some might want information about other targets. Or they might want access to your equipment, such as computers or routers, to carry out other hacks. They might even be looking for insecure devices, just to have a little fun. In these cases, any device that is connected to the internet becomes a target for black hats.

We all have plenty of devices, some we might not even know about, that use the internet and need to be secured against black hat attacks. In this chapter, we'll briefly look at how the internet works, including a history of the technology, to help you better understand how an adversary

uses it. Next, I'll break down how black hats prepare an attack with the information they gather from public resources. I'll finish the chapter by explaining how to hide from attackers by implementing three essential rules of internet use.

How the Internet Works

To comprehend how a black hat finds and exploits you on the internet, you need to understand some fundamental concepts about how the internet works. The internet as you know it today began as a project in the *Advance Research Projects Agency (ARPA)*, a United States government organization tasked with researching new technologies to maintain a lead over the Soviet Union.

In the 1960s, ARPA began working on a tool that would protect US communications during a nuclear attack. Because nuclear bombs could easily wipe out massive amounts of infrastructure, the US military needed a communications network that could reactively realign itself should part of the country be attacked. For example, if Washington, DC, was hit with a bomb, the military needed to be able to bypass the communication lines that went through the city so it could continue to share information with other parts of the country seamlessly.

One solution to this problem was the idea of *packet switching*. The premise was to put information into packets, or self-contained units, and then have a computer decide in real time where those packets should be sent based on information provided to it. For instance, if a computer received a packet destined for Atlanta (identified by an address attached to the packet) and knew that the intermediary Washington, DC, communication lines were down, it could automatically send that packet to, say, Cleveland, which could then pass it on to Atlanta. This allowed computers to create and maintain a network of communication even if part of the network was destroyed.

ARPA and many other researchers worked on implementing packet switching in large networks. Until that point, devices had communicated with each other directly via dedicated circuits set up between them. A circuit was typically a single physical line, and any break in the line would bring down the entire network. By the late 1960s and early 1970s, several smaller networks, created mostly to communicate between various universities and supercomputer sites, used packet switching to allow computers to communicate with each other over vast distances. Figure 2-1 shows a breakdown of the sites connected as part of *NSFNET*, one of the early networks that would later become the internet. This work continued into the 1980s, when commercial desktop computers become more readily available to the public.

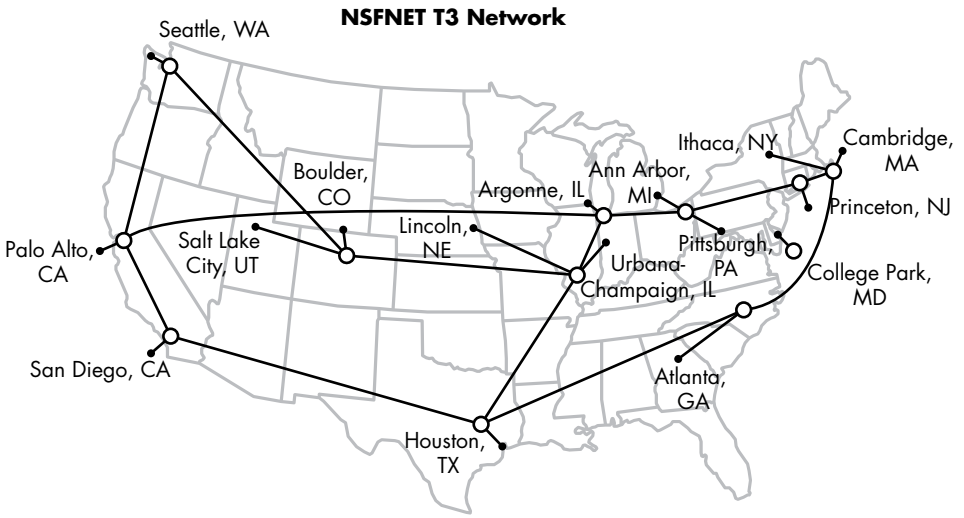


Figure 2-1: NSFNET in 1992, connecting various academic and other sites across the United States (image altered from the original created by Merit Network, Inc. under the Attribution-ShareAlike 3.0 Unported [CC BY-SA 3.0] license, https://creativecommons.org/licenses/by-sa/3.0/deed.en*)*

It was also at this time that Robert Kahn and Vinton Cerf first developed the communication protocols known as the Internet Protocol (IP) and Transmission Control Protocol (TCP).

TCP/IP: The Backbone of the Internet

TCP/IP (sometimes referred to as the *IP suite*) is the set of protocols that runs the modern internet. Protocols are special codes that define how a system should understand and process the data received over a network. For example, the HTTP protocol tells a system that the data sent is a website and should be processed by a web browser. The TCP/IP protocols tell systems how traffic (flows of data) should be passed from device to device to reach a destination. It's part of the information that systems use to make adjustments in a packet-switching network.

The IP protocol provides a number, known as an IP address, that identifies the location of a computer on a given network. You can think of an IP address as your ZIP code. A ZIP code identifies a general region that the postal service uses to direct a package. There are two versions of IP addresses, version 4 (called IPv4) and version 6 (called IPv6). In this chapter I'll only discuss IPv4, because it's still the most common.

TCP is a set of rules that allows one system to communicate with another system while ensuring that both systems are available on the network. TCP is essentially the same as calling a friend to confirm they'll be home to receive a package when it comes in the mail. We'll talk more about these two protocols in Chapter 6.

With TCP/IP, packet-switching technology, and cheaper home computers, it didn't take long before commercial companies became interested in setting up their own networks so businesses and homes could communicate. Eventually, these networks began mingling, connecting larger and larger numbers of systems until the internet evolved naturally to consist of internet service providers (ISPs). ISPs, such as AT&T, Comcast, and Verizon, began to provide internet access and sell the necessary infrastructure to businesses and eventually homes. Since the early 1990s the world has become more interconnected, with computer networks reaching nearly every corner of the planet.

Public vs. Private Networks

Today's internet is made up of a large number of connected smaller networks. These networks can generally be categorized into two types: public and private. Essentially, anyone can use a *public network*, usually by paying a fee. For example, the network that your house connects to and that you pay an ISP to use is a public network. These form the backbone of the internet, because they allow any paying customer to connect. Often, public networks are run by ISPs.

Frequently, public networks are also connected to *private networks*, which only allow connections to a limited group of devices. For example, if you work in an office, you might be able to access files from a specific server through a connection from your desktop computer. The server and the desktop computer are on a private network, meaning they're only allowed to communicate with each other or other devices on the private network. People on the public network (the internet) can't directly see, connect, or access anything on the private network.

Many private networks have a connection to the public network through equipment provided by an ISP, and they pay to get access to the internet. For example, your home might have a Wi-Fi network. Only the people who live at your house or guests to whom you give access can use that Wi-Fi network, making it a private network. However, your house is connected to the internet, usually through a special device called a *modem* or *router*. These devices pass your traffic between your home Wi-Fi network and the ISP's public internet. You, along with people in your neighborhood, pay the ISP to access the internet using special ISP equipment. Without you letting them have access, people can't just access your private network from the public network.

Figure 2-2 shows how the internet with its public and private networks might look as a visual map. Billions of *nodes* make up the internet. These nodes represent connections between IP addresses. The expanded section in the bottom-right corner of the figure reveals how individual addresses, such as 8.8.8.8, are connected to an ISP to form larger connections that create the internet.

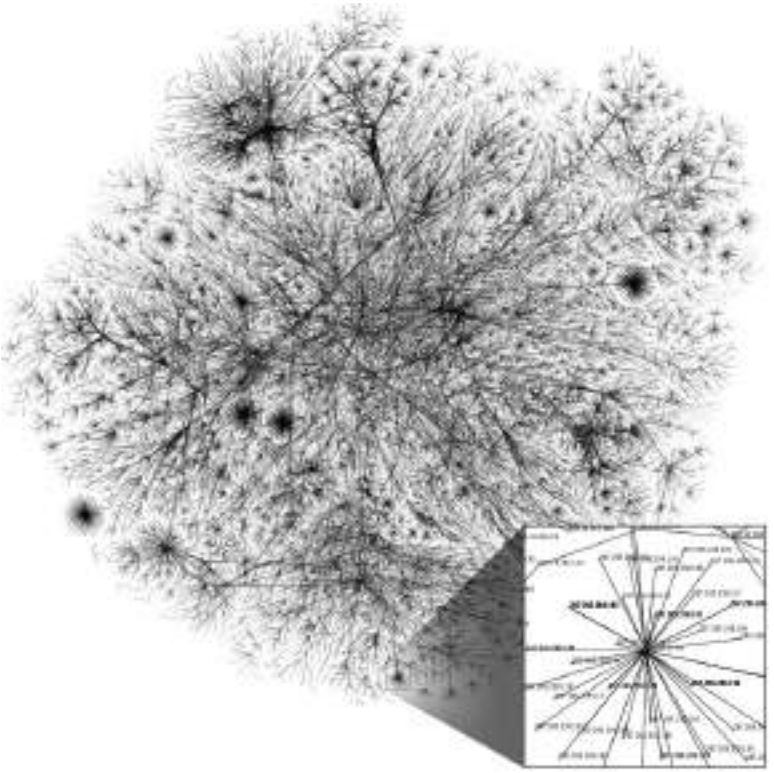


Figure 2-2: A map of the internet (image altered from the original created by the Opte Project under the Attribution 2.5 Generic [CC BY 2.5] license, https://creativecommons.org/licenses/by/2.5/deed.en*)*

Private and public networks are differentiated by the type of IP address they use. An IPv4 address is made up of four numbers, each ranging from 0 to 255, usually separated by periods: 192.168.15.1 and 10.10.10.255, for example. Certain ranges of those numbers are set aside for use by private networks only, whereas the rest are available for the public. Private addresses use certain addresses that never change, such as 10.0.0.0 or 192.168.27.0. Public addresses get translated to private addresses when you connect from a public network to a private network, and vice versa. For example, when you connect to google.com, you might connect to the address 8.8.8.8 (a public address). Once the connection to 8.8.8.8 is made, Google's own network equipment might translate that connection into a private address, such as 192.168.1.1, so you can access resources on Google's private network. This process is known as *Network Address Translation (NAT)*.

How the Internet Looks to a Black Hat

When a black hat accesses the internet, they're often trying to figure out how they can get past the public network and look into a private network.

This can be exceedingly difficult, because many of the systems that make the internet what it is today are designed specifically to prevent people on a public network from seeing what's going on in any private network. Consider the NAT process: the person connecting to the public address doesn't see the translation or what is happening behind the scenes. When you connect to google.com, you connect through a web browser, and the Google website appears. You're not informed of and normally can't see all the mechanisms on Google's private network that allow the web page to operate the way it's supposed to.

So when an adversary focuses on a target, their first step is often to determine how they can move from the public side of a network to the private side. Once they're in the private network, they can work on finding their specific target and executing the attack to get whatever they're after, whether that's disrupting business as usual or stealing data. To perform all these actions successfully, many black hats rely on a certain set of steps to maximize their attack's potential.

The Black Hat Attack Methodology

Not every attack by a black hat follows a specific pattern or set of steps. But most attackers must accomplish certain objectives before they can fully realize their goals. Several models classify these objectives, but one of the most famous is the *Lockheed Martin Cyber Kill Chain (CKC)*.

The CKC consists of seven steps that a black hat must accomplish for their attack to be effective. These steps involve activities undertaken before, during, and after many cyberattacks: they include *reconnaissance*, *weaponization*, *delivery*, *exploitation*, *installation*, *command and control*, and *attack on objectives*. Let's look at each step in more detail.

Reconnaissance

During the reconnaissance phase of the CKC, the attacker learns everything they can about their target. They begin by collecting any data considered public information. In the case of an organization, this means data from their websites and social media, as well as data about their employees, their organizational structure, physical locations, groups they've partnered with, recent news stories, public IP addresses owned by the organization, and more. For an individual, this might include information about their family members, where they work, where they live, criminal and other government records, and of course, social media.

These adversaries also look for not-so-public information, much of which is still available on the internet if you know where to look. This includes websites that, although accessible by everyone, might not be meant for public consumption, such as an employee's remote login page. Or it might include details collected about a public IP address, such as the services it's running. The black hat might also begin listing relevant email addresses by mining social media or other places on the internet to use at a later time.

One of the ways that attackers can find information is by sniffing and scanning. *Sniffing* is intercepting and analyzing other users' network traffic. Black hats can do this without interrupting the traffic flow, so the user remains unaware that their traffic is being monitored. For example, an attacker might look at all the data coming from an email server and copy any attachments before sending the original emails to their destinations. *Scanning* is sending specially crafted packets to a device and listening to how it responds to those packets. The responses can provide the black hat with information about what kind of system or software the node is running. For example, if an attacker wants to know whether a host is running a Windows operating system, they can send a packet made specifically for a Windows system. If the system responds with an error, they know it's not a Windows system. We'll talk more about scanning and sniffing in "Attacking Your Network" in Chapter 6; both can be lucrative sources of information for an adversary.

All of this reconnaissance work provides information that helps black hats narrow their field of focus until they know where to begin their initial attack. It also provides valuable information they can use in the next phase to craft attacks that are likely to work. For example, if an attacker scans a public IP address and finds it's connected to a Windows server, they won't waste time using Apple exploits on it. This is what makes reconnaissance such a key part of mounting a successful attack.

Weaponization

In the next step, weaponization, the black hat creates an actual attack to use against a target. With the information gathered from the reconnaissance phase, they plan and create the tools they'll need. This phase also requires that the adversary have a good sense of what will get them to their mission objective the quickest. For example, if the attacker's objective is to gather more personal information about a target and use it to blackmail them, they might try to exploit the target's email. Creating a virus that destroys Word documents, although an effective attack, wouldn't be a great way to meet this objective. Instead, it's a much better idea to create a PDF that links to a fake email login page and try to trick the victim into using it. That way, the attacker may be able to gather the victim's credentials from the fake page and use them to log in to the legitimate account.

Delivery

Once the black hat has a weaponized package, whether it's malware, a phishing website (which we'll discuss in more detail in Chapter 3), or some other form of attack, they're ready to deliver. Again, this requires using the information gathered during the reconnaissance phase to decide what the best method of delivery will be. Many recent attacks have been delivered through email, but this might not always be the best method.

If the attacker knows that the target uses a device with a known flaw, they might craft a delivery method that takes advantage of this flaw. For example, if a company website was using a fillable form that included exploits, an

adversary might be able to inject code directly onto the web server through the exploit. This would allow them to deliver their attack directly to the server instead of having to rely on an employee to install it for them.

Exploitation and Installation

The next two steps, exploitation and installation, rely on getting the exploit installed once it's delivered. This means getting a person to click a malicious link or launch the malware created during the delivery phase. Once the exploitation is done, the black hat should be able to execute their attack or install malware on the device.

Keep in mind that many of the items attackers want, such as credit card numbers or other personal information, are usually stored on private networks, inaccessible to the public. This means that attackers must compromise the private network before they can fully access it.

This compromise usually involves the adversary establishing *backdoors*. Consider this analogy: if the front door is the way people are supposed to enter a house, then using the back door (or garage door) would be a way for someone to bypass the controls, such as the lock on the door. Black hat backdoors work in a similar fashion, allowing an adversary to access the system without having to go through the normal, trusted means of authentication.

Command and Control, and Attack on Objectives

During the command and control and attack on objectives phases, the black hat uses the backdoor to establish a foothold in the system. From there, they can use it as a base to identify further systems to exploit. This is known as *pivoting*. Attackers will continue to pivot until they can reach their objective directly (as discussed in Chapter 1, objectives will vary depending on the type of attacker). Once they find a way to their objective, the attacker will launch a full set of attacks to gain access and accomplish their mission.

The command and control phase involves creating a command and control server, which is a tool that allows the attacker to send the compromised device commands from a remote location and receive information. For example, if a black hat compromises a web server, they might instruct that server to reach out to other devices on the network to find additional systems they could compromise. Often, these commands use normal traffic patterns to hide, so it's harder for white hats to detect them until it's too late.

The attack on the objective is usually done in a similar stealthy manner to ensure that the black hat isn't prevented from getting what they want and that the organization can't mitigate the damage. If an attacker steals a number of credit card numbers, they're only useful if the bank doesn't know they've been stolen, at which point it would cancel them before they can be sold or used. With this final phase complete, the black hat sells their prize and moves on to the next target, again starting at the reconnaissance phase.

How Black Hats Find You

If you look closely at the phases of a black hat's attack, you'll notice that one of the most important steps is the first one: reconnaissance. If an adversary can't find any useful information about their target, they'll have an extremely difficult time delivering an effective attack. This, of course, makes it that much harder to get a foothold in the private network.

So, where do black hats find their reconnaissance information? They find it mostly from publicly available sources, which people often create without realizing what they're exposing. Often, misconfigured systems openly communicate on the internet, exposing services that an organization might not want available to the public. You can see many of these open systems by using *Shodan*, a tool that scans the internet for open services and systems. After a scan, Shodan puts its findings in an easy-to-use database that is open to the public to search through. Using Shodan, you can find all sorts of detailed information on the types of devices that are publicly accessible from the internet. We'll walk through using the tool in the exercise at the end of this chapter.

Using Shodan isn't the only way to find useful information online. A ton of data on the internet might help a black hat craft an attack. Let's look at a few scenarios to help you understand how an adversary can gather this type of information.

Example 1: The Merger

Say an attacker learns that Sparkle Kitten Inc. is buying Smelly Puppy Co. and merging the company directly into Sparkle Kitten. By reading the news, the black hat learns that the CEO of Smelly Puppy is unhappy about the merger. The attacker decides to target Smelly Puppy during this stressful time. They begin by scanning Smelly Puppy's website to look for any listed email addresses. By using an automated tool to comb through all available web pages, even those that might not be available via a Google search, they locate a job ad for an administrator with knowledge of a specific type of web server.

Using public registration information, the black hat can find exactly which IP addresses the company bought and registered to use. The adversary then uses a scanning tool that targets those addresses, looking for that particular web server. They find the server, and what's more, discover that it responds to the traffic sent to it. Now they can craft an effective attack using a known exploit and gain access to the server.

Example 2: Social Media Hunting

A black hat wants to gain access to Secure Co., one of the most secure companies in the world (it says so in its name). The attacker knows that Secure Co. uses the latest appliances, training, and best practices to remain secure, because Secure Co. advertises this information often. The black hat also

realizes the organization uses a specific marketing company, Super Awesome Marketing, for all its advertising. The adversary decides that instead of attacking Secure Co. directly, they'll attack Super Awesome Marketing.

To do so, the attacker looks through LinkedIn and Facebook to find employees who work at Super Awesome Marketing. They locate a particular employee who works for the IT department and track them on Twitter. Every morning, this employee takes a picture in the same gym. The black hat also notices they leave geolocation tags on their posts. Using those tags, the black hat finds the gym the employee uses. The attacker visits that gym one morning, listens to the employee's conversations, and hears about a particular exploit in Super Awesome Marketing's email server. The adversary uses that exploit to gain access to the email server, where they can then take over an employee's email account. Now the black hat has a means of infecting the marketing material Super Awesome Marketing creates for Secure Co. Because this material is coming from a trusted vendor, it proceeds right past the normal security checks and stealthily sets up a backdoor inside Secure Co.'s private network. Not so secure now, huh?

How to Hide from Black Hats

The previous examples on how an adversary gathers information might seem far-fetched, but they describe real-life techniques that black hats have used. When people post information publicly, attackers can use it to find cracks in their security, allowing the attackers to craft the perfect attacks against a person or organization. The best way to defend against these attacks is to implement *operational security (OPSEC)*.

OPSEC is the process of understanding and minimizing any information that could be used against you. The technique originated in the military, which worried about tipping off an enemy about an attack by revealing seemingly noncritical information. For example, if the military moved a unit to a new base, an opponent could correlate this action with other information to deduce that the military was planning an attack on a certain country, perhaps one that was closer to the new base.

For civilian organizations, OPSEC is about protecting information that a black hat could use to attack your organization. This means limiting the information you share on a public website, press release, or social media. OPSEC is tricky to get right, because it's difficult to know what an attacker might find useful in the right context. The best way to ensure your OPSEC is to keep three rules about the internet in mind when posting information: the internet is open, public, and forever.

The Internet Is Open

When you're using the internet, assume that anyone can see what you're doing or sharing, including any data moving across the network. It's up to you to protect that information by determining how you send it.

A good example is the ability to request web pages. When you access a web page, your browser has to figure out where that page is located on the

internet. It does this by querying a *Domain Name Service (DNS)* server, which contains records of the public IP addresses to which websites are assigned. For example, a DNS server might tell you that the domain sparklekitten.net is at the IP address 1.1.1.1. When you tell your browser to go to sparklekitten.net, it sends out a request that eventually ends up at the DNS server, which provides the record of the IP address where you can find sparklekitten.net so your browser can access that website.

Usually, DNS requests travel through a series of DNS servers until they find the right one. Your browser starts by sending a request to a server, hosted by your ISP, which sends it to another DNS server, which sends it to another, until it finds the Sparkle Kitten DNS server with the correct record. Until recently, browsers sent these requests almost entirely unencrypted, meaning they remained in plain view for anyone to see. So not only could your ISP see every web page you requested—information it was more than willing to sell to marketing firms—but anyone with the ability to sniff your traffic could also see your DNS requests. Even if you were using a private browser or visiting websites using encrypted links, that DNS request used unencrypted protocols, so anyone could know which website you were trying to reach.

Fortunately, many browsers have since begun supporting DNS requests sent over encrypted links. Still, this is a prime example of how the internet is open. As you browse, send email, or download files, you're relaying information that is being cataloged, stored, and often sold. This information can easily be exploited to learn about you or your organization to craft the perfect attack. This is why it's important to be mindful of the kind of information you communicate on the internet. Although you don't need to cut yourself off from the world entirely and live in a cave, it's best to ensure you encrypt any sensitive information, especially if you send it through email, file sharing, or social media. It's frequently a good idea to research the services you use and what data they might be collecting on the backend. Even though it might take additional time and effort to take these steps, the extra security and peace of mind they offer is more than worth it.

The Internet Is Public

The internet is completely public; anyone can get online as long as they have the right connection set up or pay a company, like an ISP, to use their equipment. In many ways, access isn't even tied to a specific person. It's possible, legal, and often best to hide who you are on the internet by using usernames or hiding your IP address (more on this later). This applies not just to usernames on a video game or social media site, but also to your IP address and your physical location in the world.

One way to track down an IP address's location is to look up the registration information using a *Whois* search. Whois is a database of website registration information. Several websites provide Whois information, including Myip.ms, shown in Figure 2-3 displaying the Whois record for the IP address 1.1.1.1.

Figure 2-3: 1.1.1.1 Whois record

Although public IP addresses are tied to specific regions in the world, it's very difficult to trust that the person using an IP address is actually present in that location. Just like you can translate public IPs to private IPs, you can translate a public IP address to a different one. This can make it difficult to track down where traffic is actually coming from, which allows a black hat to easily hide in plain sight.

This also means that people from other countries, your teachers, your grandmother, or even your postal worker can access what you post on the internet. More importantly, if you put something on the internet and make it public, it can be difficult to stop people from seeing it. Even if you think you're only sharing something with your friends, it's possible that your friends are sharing it with the public at large. The best rule to adhere to when posting any information to the internet is to assume that everyone will be able to see it, so craft what you say with that assumption in mind. If you think the post might hurt you or provide information that others can use against you, it's best not to post it in the first place.

The Internet Is Forever

It's nearly impossible to delete information from the internet. For instance, when you delete an email, is it actually gone? If you're using a service like Gmail, deleted mail goes into a trash folder, which holds it for 30 days before it's removed from sight. So the email you deleted isn't really deleted; it's just placed in a different place where an adversary could still access it.

In the case of social media, the situation is even worse. Companies like Facebook and Google make a lot of money from the data people create on their platforms, so it benefits them to hold on to it for as long as possible. Facebook and Twitter store posts for many years. Even when you remove yourself from the platform, posts you've made in groups that you were a part of remain publicly available. Try googling your full name and state; you might be surprised to find your posts in the search results.

Also, many people document online activity to keep a record of the internet as it changes. One of the main projects doing this work is the Internet Archive, at *https://archive.org/*. The Internet Archive attempts to catalog every web page created, so even if you've removed or edited web pages, it's possible a record of them exists for people to find.

Just as it's important to assume that everyone can see what you post on the internet, it's just as important to assume that your internet posts will exist forever. Again, this doesn't mean you should forgo using the internet entirely. Just be mindful of what you do when you're online.

Understanding the three rules of the internet will help you practice OPSEC if someday you work for an organization that needs to prevent sensitive information from becoming public. By being mindful of how you post your personal information, you'll notice information that black hats could potentially use to attack your organization. You could also teach others in your organization, especially new hires, about the importance of limiting the information they share with the public. This behavior will make your organization more secure overall. After all, the less information an attacker has, the harder it is for them to attack.

Exercise: Analyzing Your Network

As you learned in this chapter, it's important to understand the information you're posting to the internet. Otherwise, attackers might use a post you've unknowingly left visible to access your accounts or your private network. As mentioned earlier, you can use Shodan to find this information, which, as you'll recall, is like a search engine for IP addresses.

Although you can use Shodan in your web browser, other useful tools require the *command line*, which allows you to enter commands on your system to perform tasks. In this exercise, you'll learn how to use some of these simple commands to discover information about your network. Then you'll use that information to search Shodan to see what sorts of services you're leaving open on the internet.

Network Command Line Tools

Windows and macOS operating systems come with built-in tools that can help you learn about your network. Let's look at four of these tools that are particularly useful for finding information that you can then use when searching with Shodan. Before you can begin using commands, you'll need to access the command line on your system. Windows and macOS have different command line programs; each uses slightly different versions of the commands and has different outputs. Let's look at them separately.

Windows

Locate the search bar in the lower-left corner of the screen and enter **CMD**. At the top of the results, you should see an app called *Command Prompt*. Select it, and a window like the one in Figure 2-4 should appear on your

screen. If you're not running as an admin, the text after C: will be your home directory and include your current username.

Figure 2-4: The Command Prompt window

First, we'll use the `ipconfig` command. This command outputs your current networking configuration, including your computer's assigned IP address, its default gateway, and information about your DNS server. The default gateway is the first router to which your computer connects to transmit traffic out of your network. Routers pass traffic from one to another to connect two endpoints together. A router creates a single network, which devices can join. So, the default gateway address is the address your computer needs to know to send traffic to the router that controls the flow of traffic into and out of your network. When you enter **`ipconfig`** in the Command Prompt window, you should see output similar to the following:

```
C:\Windows\System32> ipconfig
Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : lan
   Link-local IPv6 Address . . . . . : fe80::4d78:5074:4095:fe97%18
   IPv4 Address. . . . . . . . . . . : 192.168.86.36
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.86.1
```

```
Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Notice the section labeled `IPv4 Address` below `Wireless Lan adapter Wi-Fi`. This is your computer's private IP address. From this output, you can see that this system was assigned a private IP address. Also notice the section titled `Default Gateway` two rows below the `IPv4 Address` section. This identifies the router to which your system sends its traffic to leave the private network. In home networks, this is often the modem or router that the ISP provides. The default gateway also has a private IP address.

Although `ipconfig` gives you great information on what address your computer is using to communicate on your local network, it doesn't help you if you want to use Shodan, because you'll need to search for a public IP address, not a private one. You can use online tools to discover public IP addresses, but we'll use the `nslookup` command because it's another command line tool that is usually available. This tool looks up IP addresses assigned to website domain names. To use it, you'll need a target. For this exercise, let's use google.com. Run the command by entering **`nslookup`** in the Command Prompt window followed by **`google.com`**:

```
C:\Windows\System32> nslookup google.com
Server:  testwifi.here
Address:  192.168.86.1

Non-authoritative answer:
Name:    google.com
Addresses:  2607:f8b0:4002:c09::8b
          2607:f8b0:4002:c09::65
          172.217.9.14
```

The output from `nslookup` shows the public IP addresses currently attached to google.com. This tool is useful when you're trying to determine the IP address attached to a website to figure out where suspicious traffic in your network is coming from. Your output might look different depending on where you're located and the current configurations Google uses.

Now that you have a public IP address, you can use another tool called `ping`. This tool sends a small packet of information to an IP address and then listens for the ping's destination to respond back with its own packet of information. This tells you whether or not you can communicate with the system, because the system can't respond if it can't receive the ping in the first place. You can try using `ping` against the public IP addresses that you discovered using `nslookup`. Simply enter **`ping`** followed by the IP address you want to target:

```
C:\Windows\System32> ping 172.217.9.14
Pinging 172.217.9.14 with 32 bytes of data:
Reply from 172.217.9.14: bytes=32 time=14ms TTL=116
```

```
Reply from 172.217.9.14: bytes=32 time=14ms TTL=116
Reply from 172.217.9.14: bytes=32 time=14ms TTL=116
Reply from 172.217.9.14: bytes=32 time=15ms TTL=116

Ping statistics for 172.217.9.14:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 15ms, Average = 14ms
```

As you can see, `ping` sends out four packets. Each packet is tracked by how fast it goes out and returns to its point of origin. The speed is so fast, it's recorded in milliseconds. In this example, each packet took approximately 14 ms. At the end of the command, the system provides a summary of how many packets were sent and received. If you're unable to reach a system, `ping` will show the packets as lost.

Let's use one last tool that will provide you with all the information you need to search with Shodan. You know you can reach Google's IP address because of the results from `ping`, but they don't tell you *how* your packets actually got to Google's system. To learn that, you can use the `tracert` tool, which sends packets to each router along the path between your computer and the destination you want your traffic to reach. These packets provide information about the stops (or hops) your traffic makes on its way to its destination, using a feature called *Time to Live* (*TTL*). Essentially, each packet is designed to make only a certain number of hops based on its TTL number. A hop is counted when the packet is passed by a router. Each time a router passes the packet of traffic along, the TTL number is reduced by 1. Once its TTL reaches 0, the packet returns information about the last router to receive the packet. The packet *dies*, so the last router to hear it sends a message to the packet's next of kin, or in this case, the device that initially sent the packet. The `tracert` tool summarizes all of these hops. Enter **`tracert`** in the Command Prompt window along with a destination IP:

```
C:\Windows\System32> tracert 172.217.9.14
Tracing route to dfw28s02-in-f14.1e100.net [172.217.9.14]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  testwifi.here [192.168.86.1]
  2     3 ms     3 ms     3 ms  Address Removed by Author
  3    12 ms    14 ms    17 ms  Address Removed by Author
  4    10 ms     5 ms     5 ms  71.154.103.34
  5    29 ms    23 ms    15 ms  cr2.dlstx.ip.att.net [12.122.138.122]
  6    17 ms    14 ms    14 ms  12.123.240.25
  7    23 ms    22 ms    13 ms  12.255.10.100
  8    23 ms    23 ms    22 ms  209.85.243.95
  9    17 ms    22 ms    14 ms  108.170.231.69
 10    19 ms    22 ms    15 ms  dfw28s02-in-f14.1e100.net [172.217.9.14]

Trace complete.
```

The output shows that the first hop made is to your default gateway (in other words, your router). From there, it takes another nine hops before your

packet gets to its destination. Each hop represents a router, either on your local network or on the internet. Each hop is sent three packets to show an average of how long it took to move to that point.

Using this tool is an ideal way to determine the parts of your network or the internet where your transmissions might encounter trouble reaching a destination. It also gives you a good idea of the public IP address assigned to your computer by your ISP; this should be the first public address you see, because your traffic has to make this hop to gain access to the internet. In the previous `tracert` output, I omitted the second and third results, because they link directly to my home network. But in a normal `tracert` execution, you'd be able to see these addresses.

macOS

On macOS, open the Terminal app to access the command line. To do this, use the search bar at the top-right corner of the screen. Enter **Terminal** and click the application that appears. Now you can learn some useful commands to help you find information about your network.

On macOS, you can use commands very similar to the Windows 10 commands, although some require slight variations. For example, instead of using `ipconfig`, you'll use the `ifconfig` command on macOS. The `ifconfig` command provides the same information `ipconfig` does but with much more detail, as you can see in this output:

```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
inet 127.0.0.1 netmask 0xff000000
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
XHC20: flags=0<> mtu 0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
ether b8:e8:56:16:38:10
inet6 fe80::8ec:dd2e:36cc:b962%en0 prefixlen 64 secured scopeid 0x5
inet 192.168.86.93 netmask 0xffffff00 broadcast 192.168.86.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
ether 0a:e8:56:16:38:10
media: autoselect
status: inactive
awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1484
ether ee:57:a6:16:74:96
inet6 fe80::ec57:a6ff:fe16:7496%awdl0 prefixlen 64 scopeid 0x7
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
```

```
options=60<TSO4,TSO6>
ether 32:00:1e:74:20:00
media: autoselect <full-duplex>
status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=63<RXCSUM,TXCSUM,TSO4,TSO6>
ether 32:00:1e:74:20:00
Configuration:
id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
ipfilter disabled flags 0x2
member: en1 flags=3<LEARNING,DISCOVER>
        ifmaxaddr 0 port 8 priority 0 path cost 0
nd6 options=201<PERFORMNUD,DAD>
media: <unknown type>
status: inactive
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
inet6 fe80::b740:b05f:b952:2490%utun0 prefixlen 64 scopeid 0xa
nd6 options=201<PERFORMNUD,DAD>
utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
inet6 fe80::5d8:28d2:8ad8:65a5%utun1 prefixlen 64 scopeid 0xb
nd6 options=201<PERFORMNUD,DAD>
utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
inet6 fe80::e0b5:18ed:6a4c:a999%utun2 prefixlen 64 scopeid 0xc
nd6 options=201<PERFORMNUD,DAD>
```

The `ifconfig` command returns a huge amount of information. Within this output, it can be hard to find your device's IP address. Look for `en0` (ethernet 0), which usually identifies your main network adapter. By default, your main network adapter is assigned your IP address.

The `traceroute` command on macOS is similar to the Windows `tracert` command. It follows the same syntax, entering the command and then a target you want to trace to:

```
$ traceroute 31.13.93.35
traceroute to 31.13.93.35 (31.13.93.35), 64 hops max, 52 byte packets
 1  testwifi.here (192.168.86.1)  2.753 ms  2.391 ms  1.938 ms
 2   REDACTED 2.349 ms  2.619 ms  2.141 ms
 3  REDACTED 13.995 ms  4.940 ms  4.207 ms
 4  71.154.103.34 (71.154.103.34)  5.964 ms * *
 5  cr2.dlstx.ip.att.net (12.122.138.122)  16.537 ms  17.924 ms  20.084 ms
 6  dlstx41ome9.ip.att.net (12.123.18.177)  14.537 ms  15.603 ms  14.522 ms
 7  12.245.171.14 (12.245.171.14)  15.592 ms  17.718 ms  31.346 ms
 8  po104.psw04.dfw5.tfbnw.net (157.240.49.143)  14.118 ms  13.705 ms
    po104.psw02.dfw5.tfbnw.net (157.240.41.125)  23.049 ms
 9  157.240.36.39 (157.240.36.39)  18.651 ms
    157.240.36.135 (157.240.36.135)  17.058 ms
    157.240.36.37 (157.240.36.37)  18.979 ms
10  edge-star-mini-shv-02-dfw5.facebook.com (31.13.93.35)  14.644 ms  20.972
    ms  20.617 ms
```

The `nslookup` and `ping` commands are nearly the same on macOS as on Windows. One key difference is that on macOS, `ping` doesn't perform only four pings by default. Instead, it continuously pings a system until the user manually stops the command. This can be useful if you're changing configurations on a system and want to make sure nothing you're doing is obstructing network access. But in most cases, you'll want to limit the number of pings you send to four or five to avoid sending too many pings at once. You can set the number of pings to send using the `-c` argument, which is short for count:

```
$ ping -c 4 192.168.86.1
PING 192.168.86.1 (192.168.86.1): 56 data bytes
64 bytes from 192.168.86.1: icmp_seq=0 ttl=64 time=1.891 ms
64 bytes from 192.168.86.1: icmp_seq=1 ttl=64 time=2.907 ms
64 bytes from 192.168.86.1: icmp_seq=2 ttl=64 time=5.073 ms
64 bytes from 192.168.86.1: icmp_seq=3 ttl=64 time=9.108 ms

--- 192.168.86.1 ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.891/4.745/9.108/2.769 ms
```

Using Shodan

Shodan comes in two forms: a command line tool you can install and a website you can browse through. For the purposes of this chapter, we'll look at the website only. You can access Shodan at *https://www.shodan.io/*. At the website, you'll need to sign up for a free account. The free account allows you to use most of the tool's functions, including searching through its databases, but it limits the amount of reports and other information you can download from the website. Figure 2-5 shows the home page.

Once you have a free account, browse through the website to become familiar with its layout. Start by clicking the **Explore** tab, which is just to the right of the search bar, near the top of the page. This page provides a breakdown of the various IP addresses Shodan has in its database and the services that are exposed on those addresses.

Figure 2-5: Shodan's home page

On the left side, you should see a few interesting categories. Click the one labeled **Video Games**. You'll see a list of various online games, including *Counter Strike*, *Starbound*, and *Minecraft*. If you click Minecraft, you'll get a rundown of all the open *Minecraft* servers currently located by Shodan. Figure 2-6 gives you an example of the list.

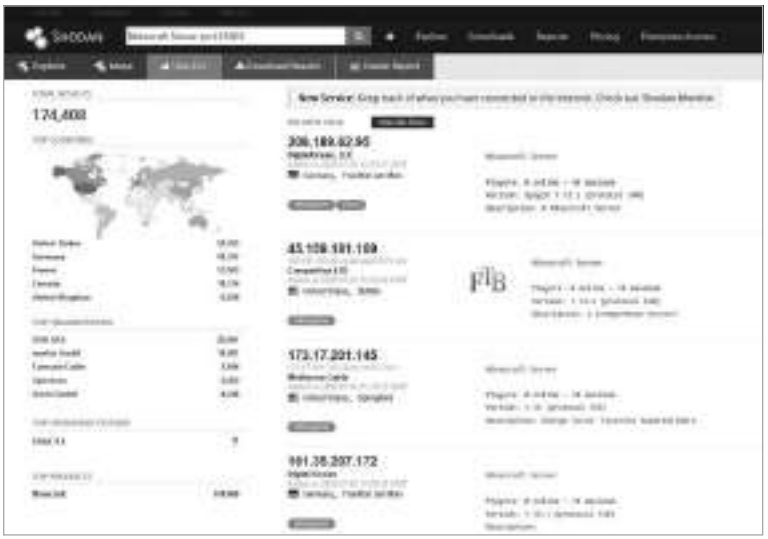


Figure 2-6: Minecraft *servers located by Shodan*

Shodan can also provide more serious information, such as information that attackers can use to exploit systems. Return to the Explore page, and this time, instead of selecting Video Games, click **Default Password**, which is about halfway down the page in a light gray box. A list of systems Shodan has verified that use default passwords for their authentication credentials appears, as shown in Figure 2-7.

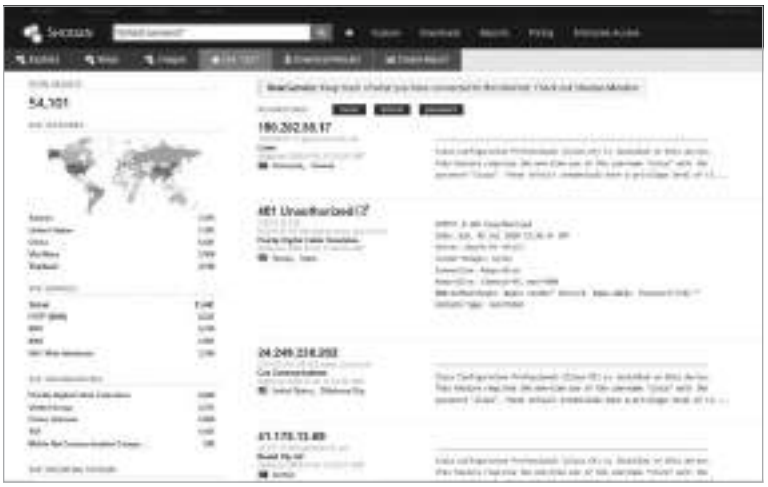


Figure 2-7: A list of systems using default passwords

Using a default password is a good way to invite a black hat into your system. Shodan allows you to check whether any of the IP addresses you're using have default credentials. It can also tell you which services you're leaving open to the internet. For example, on the left side, you'll see a list called Top Services. Click **Telnet** to pull up a list of systems that allow Telnet connections, as shown in Figure 2-8.

Figure 2-8: Systems with Telnet open

Telnet allows you to make a remote connection to a system and send it commands as if you were an administrator. It essentially lets you to control the system. Oh, and all Telnet traffic is sent unencrypted. This makes it rather dangerous. But as you can see, a ton of devices allow Telnet connections. Click the IP address in the list to see where these devices are located and other information about them. Figure 2-9 shows an example of a system in China with Telnet open.

Figure 2-9: An IP record on Shodan for a system in China

Getting paranoid yet? Shodan gives you a massive amount of information about what the public can see about a network on the internet. You can also use it to search for specific addresses. In the search bar, try plugging in the intermediary addresses you discovered during your `tracert` trial to see what comes up. You might not like the results, but at least now you know what your network is putting out for black hats to see.

The key to hiding from an attacker is first knowing what they can see. Using the command line tools in this exercise and Shodan, you can gather that data. If you find that one of your devices is exposed on Shodan, you can take a few steps to close the exposure. First, you'll need to identify which device is exposed. Shodan provides additional details about the connection that can be helpful to accomplish this. Second, you'll need to control the exposure. You have some options here. You can remove the device from the network entirely, although this often isn't viable, because it might not continue to function. You can also look up the manufacturer and find out whether it has recommendations for securing the device. If that doesn't work, forums like Reddit and Spiceworks often provide advice on how to secure home networks. Once you know which device is open, it becomes that much easier to close it.

Conclusion

The internet is a complicated construction of devices and connections that span the globe. Although this can be difficult to conceptualize, you don't need to understand every aspect of the internet to use it securely. By understanding how black hats find targets and how they create attacks against those targets, you can better defend your system when using the internet. The first step is knowing what information you're making public. Once you know what is exposed to attackers on the internet, you can better deploy your own defenses.

3

PHISHING TACTICS

Although it might not seem like it, humans are fairly predictable when it comes to certain behaviors. Black hat hackers know this and use it to their advantage using a technique called *social engineering*, which involves manipulating a person into doing something or revealing some hidden information the victim would normally not do or divulge.

Attackers use social engineering techniques to gain access to your system or data by tricking you. In this chapter, we'll discuss some social engineering techniques that attackers use to gain access to intel, including phishing, URL hijacking, and even hoaxes. By the end of this chapter, you'll have a good idea of how to spot fake messages and counterfeit websites, helping you to avoid any adversaries trying to steal your personal information.

What Is Phishing?

Phishing is one of the most common types of social engineering attacks. It's an attempt to trick a victim into revealing critical information, usually via email. Most likely, you've seen emails that start with an offer to send you a million dollars or promise a cool prize if you just click this link. You might have laughed at the terrible use of grammar or the hilarious premise as you pressed the DELETE key. These are examples of common phishing attempts.

Black hat hackers will try to appear as legitimate individuals or organizations and offer some sort of reward or present a crisis only you can solve. For example, they might pretend to be from your bank and tell you that "You need to respond with your account details before your account is locked out." By adding urgency and intimidation, they're hoping you'll be scared enough to do what they want without second-guessing their tactic.

These attempts usually look for details such as personally identifiable information (PII), credit card numbers, or passwords for important online accounts like your bank or email account. Sometimes they ask for this information directly in the email. Often, they'll ask you to click a link to a website that mimics a real website but is actually a malicious site that will steal and record any information you enter into it, such as your password and username. This is a slight variation of phishing known as *pharming*. We'll talk more about this in "How Black Hats Trick You with URLs."

An Obvious Phish

Sometimes phishing emails are easy to spot and are automatically filtered by your email's spam settings. Let's look at an example of a typical phishing email you might find in your spam folder on any given day:

> Dear Human Greg,
>
> Itz come to our attentionz that you credit card is not update in our database. We has new system that require you to put your infomationz in again. You see, Don spilled a big cup of coffee on the last systemz. I tellz Don, NO YOUZ CANT HAZ COFFEE IN SYSTEM PLACE but he sayz I HAZ COFFEE WHEREVERS. Please, I can haz credit card number? K THX BAI
>
> Sincerely,
>
> Janice, a realz human. (NOT CAT)

This email obviously wasn't sent by an actual person named Janice. It has numerous grammatical errors and includes unprofessional language. It also doesn't mention what service they represent, let alone why they would send you an email directly to update your information rather than having you log into a personal account (which is the typical practice). Additionally, many details are included that would be unnecessary for an account update

email. Often, phishing emails will include a narrative intended to get you to trust or empathize with the sender, such as a story about being deported from their country or having recently lost a loved one. The details are provided to confuse or trick you.

Not All Phishing Is Obvious

Not all phishing emails are easy to identify. Let's say you received this email from *customerservice@amazon.org*.

> Dear valued customer,
>
> Your account at <insert your email address> was recently flagged for suspicious activity. Because of this activity, we've temporarily suspended your account and will be permanently deleting it in ten days if you do not verify your information.
>
> To verify your account, please click the link: <malicious link here>. This is an automated message. Please send all replies to accounts@sparklekitten.net.
>
> Sincerely,
>
> Customer Service

This phishing attempt is much trickier to spot. The phisher made sure to write a short, coherent message. It gets to the point—your account is suspended and might be deleted—and uses the social engineering principle of urgency to get you to click a link that will surely take you to some sort of malicious website or even download malware. Frequently, black hats will steal a real company logo to make their emails look more authentic. The preceding example email might have the Amazon or PayPal logo pasted at the top so you assume the email came from one of those companies.

The only real indication that this is a phishing attempt is the email address, *customerservice@amazon.org*. Often, when a phishing attempt is received, it will come from an address that is close to but not quite what the actual company might use. Usually, it contains added words or misspellings, such as *accounts@amzon.com*. If you're unsure about an email, you can always compare the address to other emails you've received from that company to see whether the domains are the same (the domain is the text that comes after the @ symbol).

Using Details for a More Convincing Phish

Sometimes an attacker will target a specific person or organization to gain access to particular data they're trying to steal, so they'll use a technique known as *spear phishing*. Spear phishing uses real information about a

person to create an email that looks so authentic it might fool even the best white hat hackers. Let's look at an example:

> Good morning Karen!
>
> This is Steve from the IT Helpdesk. How's everything in HR today? We are supposed to run updates later tonight on your system but I need to make a few changes from your account before I can do that. Can you send me your account login? I'm really swamped down here and don't have time to walk three floors to your office so I was hoping to remote in real quick. Thanks!
>
> Steve
>
> ABC Company
>
> 123 Street
>
> Anywhere, USA

The black hat really did their research for this one. They not only found someone to target who works in HR, Karen, but also found an IT Helpdesk person to impersonate, Steve. By adding little details, like the fact that HR is three floors away from IT, the attacker is able to create trust and familiarity with Karen, which are two more powerful social engineering principles.

Vishing and Other Non-Email Phishing

Email isn't the only way adversaries try to target victims. Phishing can come through any media that allows for communication between people. Instances of phishing attempts have been found in chat apps like Discord, on social media platforms like Instagram and Twitter, and even in games like *League of Legends* or *Fortnite*.

They can also use your phone. A phishing attempt using a phone call is known as *vishing* and can be especially dangerous because the person can react to you in real time. If you sound skeptical or uninterested, the black hat can change their tactics to try to entice you to give them what they want. Frequently, vishing attempts will also impersonate sources of authority, such as the police or the IRS. Imitating authority is a social engineering principle. People have a tendency to immediately trust known authority figures, like a doctor, so it's often advantageous for the attacker to assume such as role.

How to Protect Yourself Against Phishing

It might be easy for you to spot phishing emails now that you know what to look for, but not everyone understands how to spot these attacks. Think about an older relative or loved one, like a grandparent, who might not

know the telltale signs of black hat phishing. It's important to help them recognize when an adversary is attacking them, by keeping these common characteristics of phishing emails in mind:

- Phishing emails usually have some sense of urgency or authority involved. If the email says you need to do something immediately or there will be trouble, there's a good chance it's a phish.
- Be sure to check for misspellings, incorrect company logos, or weird email addresses.
- If you've never used a service, it's highly unlikely they'll email you out of the blue. You're not going to get money from a bank at which you don't have an account.
- Tech support will never call you first.
- Always go to the website rather than clicking a link in an email unless you're absolutely sure you know where the email came from.

Teaching your friends and family to consider these details when using email can help them stay safe. You can also create custom rules in their spam filters that will help guard them from common types of phishing. For example, if you know they only use Facebook, you might create a rule that sends any emails from other social media platforms to the spam folder. This will help reduce the amount of phishing emails they have to deal with, making it easier to catch the ones that get through.

How Black Hats Trick You with URLs

Many phishing emails don't just ask you straightforwardly for your information; instead, they'll tell you to click a URL that directs you to a malicious web page where a black hat hacker can harvest your passwords or even install malware on your computer. When you, the victim, click the link, you'll think you're being directed to a perfectly safe web page, so you're likely to enter your important information without a second thought.

A URL, or uniform resource locator, is an address used to find a website, such as *https://www.google.com/* or *https://www.instagram.com/*. When you enter that address into your browser, your computer reads it and sends out a Domain Name System (DNS) query, which looks for the IP address associated with that URL. It's similar to your school finding your home address by looking up your name in the school database. Essentially, this is what the DNS does for your web browser: it uses the name (URL) of the website to look up its address (IP address) so the browser can deliver the right web page to you. The DNS is held on a server, either on your local network or in many cases run by your (ISP).

Typosquatting

We use URLs so much that most people don't even pay attention to the web address anymore. That's exactly what attackers are hoping for. Black hats can

create their own URLs and use those instead of legitimate URLs to get you to go to malicious web pages. This is known as pharming.

Adversaries accomplish pharming by modifying the content in a URL or on a website. When a black hat misspells a URL, it's known as *typosquatting*. For example, they might register petmart.com instead of petsmart.com. The DNS then looks up the misspelled URL instead of the real one and sends you to the unsafe website. Today, typosquatting is a rare occurrence because many companies register every possible misspelling of their website name to ensure they all go to the same authentic website.

Complex URLs and Redirects

Black hats also create complex URLs that are hard to read. They do this by creating a long path after the initial URL. A path is where a file is found on a website. For example, *sparklekitten.net/kittenpics* would be the path that accesses the kitten pics section of the sparklekitten.net website. Attackers can use this to their advantage by creating long paths that make it difficult to see where the URL is actually going. For instance, you might get an email with a link that looks like this: *www.accounts.com/user/payments/…* with the three dots indicating that the rest of the URL was cut off. Although this might look like a valid website, there could be a more dangerous portion at the end of the path, such as *payments/files/virus.exe*.

Black hats might also use redirects to hide where their URL goes. A *redirect* is a piece of code that, when activated, sends you to another website instead of the original one you clicked. You might see an ad on a web page that shows a cool new browser game called *Cat Attack!* The ad will look authentic, but as soon as you click the ad banner, instead of going to a cool web game, a script embedded in that web page activates and redirects you to *sparklekitten.net/dumbhooman*.

Redirects are a favorite of adversaries because they're difficult to detect before a person activates them. It's also possible to place scripts and even redirects in real, legitimate websites if that valid website isn't secure (more on redirects in "Web Application Attacks" in Chapter 7).

Modifying DNS Records

Another way that attackers like to pharm is by tampering with DNS records. A DNS server uses records to organize and manage all of the websites and their IP addresses. These records are maintained across all the DNS servers on the internet, so if your DNS server doesn't have a record, it sends out a request to another DNS server until it finds what you're looking for.

If the black hat hacker can modify the DNS record, they can tell your web browser to go wherever they want. They do this by breaking into the DNS server and modifying the record there, causing anyone who queries that server to get the malicious record. Fortunately, altering DNS servers is difficult because they're challenging to break into.

Another pharming technique is to add information to your computer's *local host file*. All computers have a local host file on their system. Any DNS record added to the file will be used instead of sending out a query to a DNS server to find one. If an attacker gets access to that file, they can create their own records. As with modifying DNS records on the server, accessing the local host file is difficult to do.

A much easier way for adversaries to attack your system is to change where your DNS queries go. Instead of them going to the correct DNS server, the black hat can make them go to their malicious DNS server. This is either done locally on your computer or, more often, on a router that your data passes through. Because your system accepts the first record it receives, the attacker can redirect all your internet traffic using their deceptive DNS records. If this happens, not only will links be directed to an unsafe site, but even if you enter *www.facebook.com*, you will still be sent to a dangerous site. The creation of a fake DNS server or record is hard to detect and is currently a hot topic among cybersecurity researchers.

Hoaxes

A *hoax* is a made-up story created to spread false information about a particular subject; for example, on the internet, it could be a fake celebrity story or a new miracle health cure. Hoaxes are initiated for a number of different reasons. Sometimes they're crafted simply as a joke, such as a ruse about new features on the latest iPhone model that don't actually exist.

Hoaxes are also created to damage or spread misleading information about a particular target. For example, a black hat hacker might be angry that a certain cat food company is no longer making their cat's favorite crunchy flavor. Using false reports of health code violations, that adversary might invent a hoax that the company's food is poisonous, thus making people hesitant to buy it.

Most hoaxes are spread through social media. A post or article containing the hoax can quickly spread via Facebook or Twitter posts. Sometimes such deceptions use real information to make them seem more legitimate, which is the reason it can be challenging to expose a hoax and disseminate the right information. Without knowing what is true, it's hard to refute the hoax, especially if it comes from someone you trust.

Deceptiveness can be a powerful weapon. With social media, it's easy for hackers to quickly publicize misinformation about a subject. This can have a huge impact, leading to distrust, anger, and confusion as people find it harder to know what is true. As a large-scale example of such dishonesty, we can look at the 2016 United States presidential election. Several false stories and hoaxes were generated about both candidates, which led to lots of misinformation being spread among the public. Any hoax has the potential to cause harm to people, so we need to always be prepared to recognize one when it appears in our social media feeds.

Why Black Hats Love Phishing

Why do black hat hackers love to use phishing techniques, including URL hijacking and hoaxes, to attack people? Keep in mind that attackers are lazy. Phishing is enticing because it's cheap, easy, and fast.

Phishing attacks are inexpensive to run because all you need is an email server to send messages. Plenty of places will let you rent an email server for very little cost. Even better, instead of paying for their own email server, adversaries might take control of someone else's. This way, not only do they get new email addresses to target from the contact lists on the server, but they can also use that system to send email, making it harder to trace the origin of the phishing messages. Even if only one person in a thousand responds, they're still likely to make a profit.

It's also incredibly easy to set up a phishing email campaign. All the attacker needs to do is craft a generic phishing email and schedule it to send at a certain time. Because phishing isn't time-sensitive, they can just wait until someone clicks the link while they move on to other projects. (Techniques like spear phishing add more complexity to the initial email because it requires custom details about the victim.)

Email is a fast medium; once the email schedule is made, hundreds of thousands of phishing emails can be easily sent in a day. This gives attackers the maximum chance of finding a gullible target in a relatively short time. Once someone clicks or replies, the attacker should have everything they need to exploit their victim.

The biggest reason that black hat hackers love phishing is that it works. It's very difficult to defend against phishing because no hardware or software can fully prevent an attack. Even spam filters miss messages. The likelihood that a spam filter will detect spear phishing is also slim. The only consistent defense against phishing is the person who's being attacked.

Think Twice to Avoid Phishing

Although it might seem as though you always need to be looking over your shoulder for phishing attempts, the best way to stay alert is to question whether an email or phone call makes sense. Doing so will help you recognize an attack.

By stopping to think about what an email is asking or a person on the phone is telling you to do, you can easily identify inconsistencies or gaps in their story. Here are a few critical details to keep in mind when you're questioning a potential assault:

- No company, no matter what, will ever ask you for your password. It might ask you to reset your password but will never ask you for it directly.
- No one ever legitimately contacts you out of the blue, especially to give you something.

- If you're told you have to take action *right now*, step back and think about whether you should do it at all.
- Legal matters, especially criminal, are rarely if ever handled over the phone or through email. Also, you should never pay a fine (for example, a tax fee or criminal fine) without first checking, in person if possible, that it's an official charge.

Take an Alternate Route

Even if you take precautions, it can be tough to recognize when someone is trying to scam you, especially if they're deploying spear phishing tactics. But keep in mind that you always have the option to use another route to check whether something is on the up and up. For example, let's say someone claiming to be from your bank calls and says there's a problem with your account. Instead of dealing with it right then, tell them you're busy and will call back later to fix it. Black hat hackers hate when this happens because they know you won't call them back but will instead call the real bank.

You can use this tactic for any phishing method. Instead of clicking a link sent to you in an email, you can go to the website by searching Google or typing in its URL directly. In fact, you should never click a link in an email unless you're absolutely sure where the email came from. You can also use well-known DNS servers to make sure you're accessing the real site. Changing your browser to use DNS server 8.8.8.8 (Google's DNS) or 1.1.1.1 (Cloudflare's secure DNS) is a good way to avoid DNS hijacking.

Listen to Your Spidey Sense

Don't ever forget that *you* are the best line of defense against phishing attempts. If you see something suspicious, listen to your inner voice and do some research to determine whether it's legitimate. It's also up to you to alert other people about it. Checking whether a source is trustworthy takes extra time, but it helps to prevent false claims from running rampant across the internet.

Exercise: Analyzing a Phishing Email

Part of being skilled at cybersecurity isn't just recognizing a threat, it's also understanding how that threat might hurt you or your organization. This is especially true when it comes to phishing emails. Recognizing certain phishing emails can be challenging. But even if you do recognize and delete one, knowing you've found a phish doesn't provide you with insight about the tricks adversaries use. Instead, when you receive an extremely well-crafted phishing email, you can use your knowledge to detect and analyze it.

In this exercise, you'll learn how to analyze a phishing email to identify where it came from, whether it's malicious, and what type of attack the black hat was attempting. By the end, you'll know some of the tricks that attackers

use to create convincing phishing emails and how to use free online tools to determine whether or not an email is dangerous.

This exercise uses the Gmail platform for its examples. But the information gathered in each step is the same regardless of what type of email application you're using.

WARNING *Analyzing phishing emails can be dangerous.* Under no circumstances should you click a link or open an attachment from a suspected phishing email. *Right-clicking the link will let you copy the link location to use for analysis without activating the link.*

Phishing Email Indicators

First, you'll need a phishing email to analyze. Figure 3-1 shows a screenshot of one I received that attempted to impersonate an Apple iCloud login warning. You can usually find a phishing email in your spam folder. Just don't download anything or click any links.

Figure 3-1: An example of a phishing email

This email purports to be from Apple and claims my account was suspended because of a suspicious login from a Linux operating system. To fix this problem, it says I just need to log in to my account by clicking the link.

This is an incredibly authentic-looking phishing email that closely mimics actual Apple emails. For comparison, Figure 3-2 shows a screenshot of a real Apple iCloud login notice.

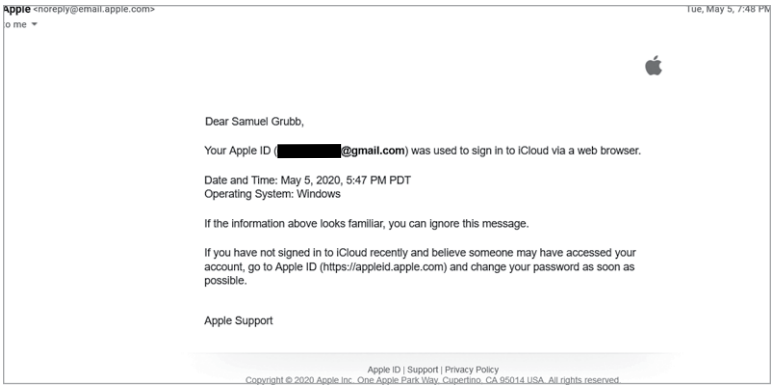


Figure 3-2: A legitimate email from Apple

It looks pretty much the same, right? So, how did I know that Figure 3-1 was a phish? Let's look at it again with a few annotations (Figure 3-3).

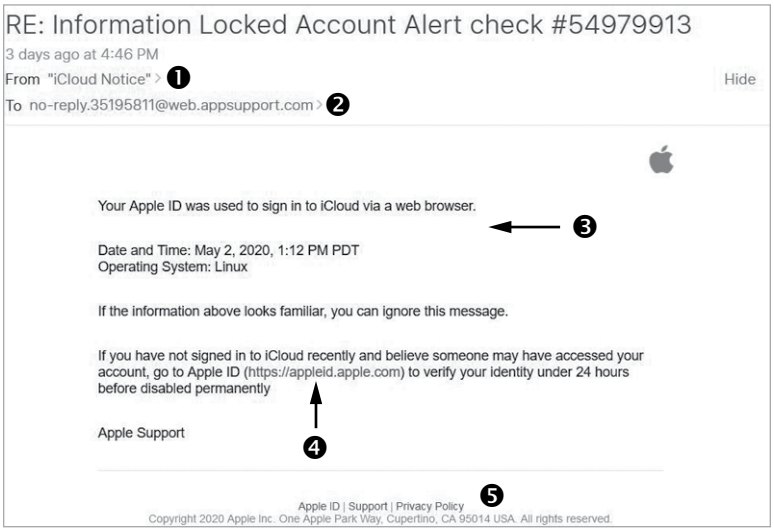


Figure 3-3: The phishing email with numbered annotations

Here is an explanation of these revealing indicators:

1. The email's sender is iCloud Notice, which is suspicious, because you'd probably expect it to just show Apple. Also, it's in quotation marks, which indicates that it's a *friendly name*. Email applications use friendly names as shorthand for email addresses. For example, if your friend Jane has the email address *sparklekittenisamazingdazzle@emaildomain.com*, the application might replace it with the name *Jane* to help you recognize the sender. Later in this exercise you'll see how black hats use this feature all the time to attempt to trick people.

2. The "To" field doesn't include my email address. This indicates that the email was sent using BCC, which hides who the email was sent to. Adversaries use this trick to send phishing emails to multiple victims without tipping them off.
3. The body doesn't contain my account name anywhere. If this alert is supposedly addressed to me, shouldn't my account name be listed? Also, there are numerous grammatical errors throughout the email, including in the last sentence. And the email tries to scare me by claiming my account will be disabled.
4. The link provided is the same as in the legitimate Apple notice, but in this email, it's *active* (clickable). More importantly, when I hover over the link it shows that the URL is something other than what is written, so it's not actually a link to Apple's website.
5. At the bottom of the email are three "links" for Apple ID, Support, and the Privacy Policy. This is probably the hardest indicator to notice. However, when I hover over them, my mouse doesn't give me the telltale hand icon indicating they're clickable links. The reason is that they aren't links at all, but just an image to mimic the signature from the legitimate email.

As you can see, even if the email is very well crafted, there are still several hints that make it apparent it's a phish. But identifying an email as a phish is only the first step of good analysis. The next part is to learn as much as we can about the email by analyzing the header and URL.

Why is analyzing phishing emails important? Let's say you're working in IT for Sparkle Kitten Inc., and a user calls in saying they received an email but are unsure whether it's legitimate. You might look at the email, realize it's spam, and tell the user to delete it. That's not a bad plan, but what if other users received the same email? What if one of them clicked the link? By taking time to analyze who sent the email and what the URL does when clicked, you'll have valuable information to pass on to your email administrator or security person should this become a problem.

Header Analysis

You'll need to analyze the header first, so you can detect where the email came from and determine any other useful information about it. The email header provides details about the email's origins (as in the stops it took to get to your inbox), who sent it, and other specific information that is included for email servers to read and use.

The process for finding the full email header differs depending on the email application you're using. In Gmail, click the three dots in the top-right corner of the email to access the menu, as shown in Figure 3-4.

Figure 3-4: The email menu in Gmail

In this menu, click **Show original**, as highlighted in Figure 3-4. Doing so will open the email in a new window and provide the full headers in a box below the original To and From fields, as shown in Figure 3-5.

Figure 3-5: Email header in Gmail

This plethora of raw data can be hard to read and understand, especially given the number of fields it contains. How do you make sense of such complicated text? You use a tool designed to read the data, of course! The first tool you'll use in your analysis is MX Toolbox. You'll find it free online at *https://mxtoolbox.com/*. MX Toolbox offers a variety of tools to use in email analysis. For now, we'll use the Analyze Headers tool. You'll see it as one of the options on the website's home page (Figure 3-6).

Figure 3-6: MX Toolbox Analyze Headers tool

To use the Analyze Headers tool, just copy and paste the full header into the blank window. The tool analyzes the header and separates all the data into easy-to-read fields, as shown in Figure 3-7.

Header Name	Header Value
Return path	<yanlodiscordolaksroelp21@ablienyx.com>
Original-recipient	rfc822; [redacted]@icloud.com
X Apple Move To Folder	INBOX
X-Apple-Action	MOVE_TO_FOLDER:INBOX
X-Apple-UUID	950e8d69-4945-449f-9199-d8ed0d8590c4
Authentication-Results	mgeni12l2.usspk05.pie.apple.com; dmarc=none header.from=ablienyx.com
x-dmarc-info	pass=none; dmarc-policy=(nopolicy); s=u0; d=u0
x dmarc policy	none

Figure 3-7: MX Toolbox email header analysis

Before we look at the header fields, we need to examine the findings that appear below the `x-dmarc-info` heading. These are related to two types of authentication that emails use: Sender Policy Framework (SPF) and Domain Keys Identified Mail (DKIM) records, which, together, are known as Domain Message Authentication Reporting (DMARC). Email applications essentially use SPF and DKIM records to verify that the email had permission to be sent from that domain and IP address. For example, if Google sends you an email, it comes from a Google email server with a specific IP address. The address corresponds to a DKIM and SPF record. Your email server checks the DKIM record for Google when it receives the email. If a black hat tries to impersonate Google when sending an email, your server will find that the IP address the attacker is using isn't the same as the one registered to Google. Therefore, the DKIM record will display as failed in the header, as shown in Figure 3-8.

Figure 3-8: DMARC failure

Although failed SPF or DKIM records are great indicators of phishing emails, they're not proof. The email server needs to have both DMARC records set up correctly for the signature system to work, and many don't. It's also possible to impersonate an IP to pass a DMARC check, so the fact that an email passes the check doesn't mean it's an authentic email.

Now let's look at the header fields. In Figure 3-9, notice that the address in the `Return-path` field at the top is `yantodiscordolaksroelp21@abtrenyx.com`, which isn't even close to one Apple would use. This address indicates that we're looking at a phishing email. Also, it's best to note the address so an email administrator can look it up later to see whether other users received the email.

Header Name	Header Value
Return-path	<yantodiscordolaksroelp21@abtrenyx.com>
Original-recipient	rfc822; [redacted]@icloud.com
X-Apple-Mobile-Folder	INBOX
X-Apple-Action	MOVE_TO_FOLDER/INBOX
X-Apple-UUID	950e8d69-4945-449f-9198-d8ad0d8590c4
Authentication-Results	mageni1202.usspk05.pie.apple.com; dmarc=none header.from=abtrenyx.com
x-dmarc-info	pass=none; dmarc-policy=(nopolicy); s=u0; d=u0
x-dmarc-policy	none

Figure 3-9: Header with highlighted fields

Moving down the list of headers, notice the headers that begin with `X`. The `X` headers hold information that the email server reads to decide how to send the email. For example, the `X-Apple-Action` header reads `MOVE_TO_FOLDER/INBOX`. This means that when the email comes into my Gmail account, it's automatically sent to my inbox instead of to junk or spam. Below these headers, you see information about DMARC. As you can see, there is no DMARC policy, which is why the email failed its DMARC check.

Table 3-1 lists some other headers to look for and the information you can gather from them.

Table 3-1: Important Email Header Fields

Field	Purpose
`Message-ID`	Unique ID given to the email. Makes it easy to find using search functions.
`x-originating-ip`	Original IP address that sent the email. Helps determine whether or not the sender was known as malicious, as well as find other messages sent by that sender.
`X-Mailer`	Specifies the application used to send the email. Weird or unexpected platforms might indicate a phish.
`Received-SPF`	Provides results of SPF check.
`X-MS-Has-Attach`	Indicates whether or not the email had an attachment.

URL Analysis

After looking at the headers, you need to verify whether the URL is malicious. To do this, you'll use another online tool called VirusTotal, which is available at *https://www.virustotal.com/gui/home/url/*. Figure 3-10 shows its home page.

Figure 3-10: The VirusTotal home page

VirusTotal allows you to scan URL links for malicious behavior by using a multitude of antivirus engines, which we'll discuss in more detail in Chapter 4. It runs the link through each engine and aggregates the information on a page that's easy to understand and share. If even one

engine flags it as malicious, you should assume the link is malicious. Figure 3-11 shows the results of running the link in Figure 3-2 through VirusTotal.

Figure 3-11: Analysis from VirusTotal

Even though only one engine returned positive malicious results, it's enough to know that this link is bad.

Like any good security expert, you have a burning curiosity to know what happens when you click the link. However, you also know that clicking the link could potentially infect your computer. So what do you do?

You use another tool called Joe Sandbox (*https://www.joesandbox.com/*). This is a free tool that lets you run attachments or open URLs in a sandbox environment. *Sandboxes* are simulated computers meant to act like real, physical machines, but you can isolate them from the rest of your computer system and destroy them easily. This makes them perfect for testing malicious entities like malware, because you can study the malware infection without worrying that it will spread or damage critical system components.

To begin using Joe Sandbox, create an account. Then copy and paste a link into the sandbox, as shown in Figure 3-12.

Figure 3-12: Joe Sandbox's home page

NOTE *Unless you pay for a private account, all results from the sandbox will be made public for other researchers to see. Don't submit anything that might contain personal information.*

It takes a few minutes for the report to generate, but once it does, you'll be provided with a wealth of information about the link and what runs when you click it. The two most interesting features are the Behavior Graph and Screenshots section.

The Behavior Graph (Figure 3-13) shows all the processes that happen when someone clicks the link, such as anything that opens or any web pages that are accessed. In this example, the link opens a few different web pages and then redirects to additional ones. You can tell that none of these are actual Apple domains, which is further proof that this email didn't come from a valid Apple source.

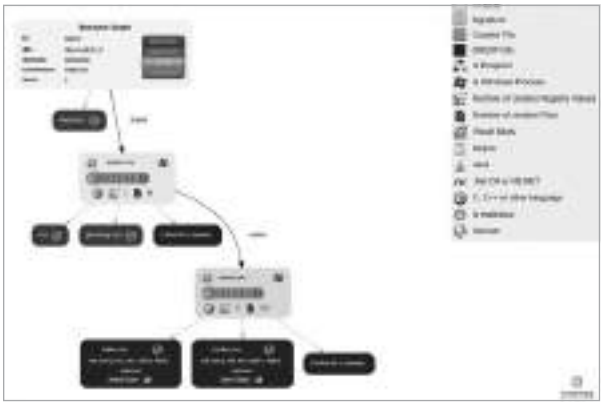


Figure 3-13: Joe Sandbox report: Behavior Graph

The Screenshots section (Figure 3-14) shows screenshots of what opened or ran when the sandbox executed the link.

Figure 3-14: Joe Sandbox report: Screenshots section

This section also has an animation option, so you can watch what happens in real time. The particular link I submitted couldn't be found, which, although unfortunate for our research purposes, isn't surprising. Phishing links typically remain active for only a limited period of time before they're either discovered or removed by the phisher to avoid detection. Still, because the email asked you to verify your account, you now know that this was likely a *credential hijacking* attack. In this type of attack, an adversary attempts to steal credentials, either by having the victim enter them in a fake site or by using browser vulnerabilities to capture them.

With a little research and a few free tools, you can learn a lot about phishing emails. You've now analyzed this email and determined that it's a phish, the source of the attack, and what type of attack was attempted. You can now better protect yourself by adding rules to your email program that instruct the server to send any message from this malicious sender directly to your junk box or to pass this information to the appropriate administrators to use in their defense efforts.

Conclusion

When it comes to phishing, it's critical to remember that it only takes one click for an attacker to gain access to your computer or potentially steal your personal information. Phishing can come from many different directions because basically an attacker can use any form of social engineering communication. Be on the alert every time you use email or receive a phone call. With practice, you'll learn how to recognize phishing attempts more easily. Whether the attack uses pharming, vishing, spear phishing, or any other type of social engineering, take time to think through what is being asked of you. Doing so can be the difference between a successful and unsuccessful phishing attack.

4

MALWARE INFECTIONS

One of the most well-known types of computer attacks is the transmission of malware. Sometimes mistakenly referred to as a virus, *malware*, or malicious software, is any piece of software designed to bypass a system's intended operation. This activity is typically not authorized by or even visible to the user. Malware has been around for as long as modern computers have. It comes in many forms, and despite the best efforts of antivirus software, it's still common today.

In this chapter, we'll discuss what malware is, some of its more popular variants, and how best to defend against it, while clearing up any misconceptions you might have picked up from watching fictional hackers on television.

What Is Malware?

Malware is software designed to cause damage to computer systems. So, even though a game might consume all of your computer's memory, it's not considered malware. The best way to define malware damage is damage caused by an unauthorized action considered abnormal for the system. For example, a normal operation might involve a user logging into the system using a username and password set up by an internal administrator. If an application allows a black hat to access the system without using a username and password, it has performed an unauthorized action.

This might seem like splitting hairs, but it's important to understand the difference between malware and a bug or other corrupt software. If a piece of software, like the game mentioned earlier, has an unintentional bug that causes a computer to crash or does other harm, it's not malware, it's just an inferior program. Likewise, a browser add-on with a privacy statement that declares, "We're going to steal your browser history and sell it" isn't malware, even if the reader doesn't bother to read the statement. On the other hand, if a program appears to be running normally while also performing hidden actions, such as logging keystrokes without informing users, it's probably malware.

For the most part, malware is fairly easy to identify because it executes obviously malicious actions, such as stealing your password or allowing another unauthorized system to access your computer. But some programs perform legitimate, authorized functions while also doing undesirable activities, such as displaying ads or recording user data. Just because a program looks, acts, and sounds like a safe program doesn't mean it isn't running malicious code in the background.

Types of Malware

To classify different types of malware, malware analysts generally use two attributes: how the malware infects a system and what sort of attack the malware performs. These attributes help put malware into broad classes that we can use to better defend against it. In this section, we'll examine some of the more common classes.

Although the malware we'll discuss in this chapter fits neatly into the classes described in the following subsections, that's not always the case in the real world. Black hats often group malware classes together to form a single malware package. For example, you might have a virus that also installs spyware and a rootkit. For this reason, it's important to scan every part of your computer when you're trying to get rid of a malware infection. For instance, just because you clean up the virus files doesn't mean you've eliminated the entire infection.

Viruses

Perhaps the most recognized type of malware is the virus, which we define not by what it does but by how it behaves. A user must interact with a virus

before it can begin executing its malicious code. We call this interaction the *trigger*. The trigger could be clicking a file, running a program, or opening an attachment. Once the action is taken, the virus can run its instructions and release its payload.

A virus's *payload* is the code that performs whatever malicious action the virus was programmed to execute. For example, many viruses created in the 1990s, such as Chernobyl, were designed to destroy infected systems, often by rewriting or deleting critical files (Figure 4-1). Chernobyl was created by Chen Ing-hau, a Taiwanese university student who wanted to prove that antivirus software of the time was ineffective. Once activated, the code rewrote the first kilobyte of the system's hard drives with zeros. This destroyed many critical files necessary for the system to function, including the partition table, which helps find where information is physically stored on a hard drive.

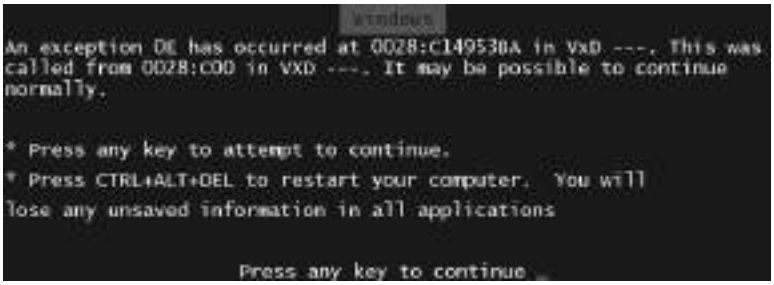


Figure 4-1: A Windows blue screen (error screen) after a successful infection by the Chernobyl virus (original image covered by the Free Art License 1.3)

As cybercrime has become more common, attackers have begun using viruses to implant other types of malware, such as trojans or ransomware (discussed on page 59), that they can then use to further exploit the system.

Although all viruses share these common characteristics, a virus's specific target might vary. For example, *file infector viruses* infect files, whereas *boot sector viruses* infect the boot files used to start the computer when you turn it on. *Macro viruses* focus on, you guessed it, macros on the system. A macro is code that translates instructions sent to the computer into a longer set of instructions. For example, when you press CTRL-C, a macro translates that keypress into the Copy command. Macros are often used in Microsoft Office applications to provide additional functionality, especially in spreadsheet applications like Excel. Some viruses, known as *stealth viruses*, include an added layer of coding to try and hide from antivirus software.

Worms

Worms are malware designed with one goal in mind: to reach as many systems as possible. Unlike a virus, a *worm* can infect systems without any direct user contact, which means it can spread through a network more easily than a virus can. Once the worm infects a new system, it looks for additional uninfected computers to which it can spread.

Worms are usually able to spread without user contact because they exploit a vulnerability that allows for *remote code execution*, or the execution of any code, even by unauthorized users, on the system from a remote location. This allows adversaries to install programs, create users, or even change network settings. Typically, the vulnerabilities that allow remote code execution involve tricking the system into believing a user or process has permission to run code. A classic example of this is the ILOVEYOU worm (Figure 4-2). Created in 2000, this worm took advantage of a vulnerability on Windows systems at the time that caused the file type *.vsb* (which indicates a Visual Basic script) to be hidden. This meant when the file was sent via email, the attachment looked like a normal text document. But when users opened it, it ran a script that infected the system by overwriting certain file types. It then used the target's email account to send copies of the file to every contact in the address book associated with Microsoft Outlook. The most common means of spreading a worm to a new system is through email. This provides the worm with a method of reaching other users and a list of email addresses with which to do so. Within 10 days, 50 million ILOVEYOU infections were reported.

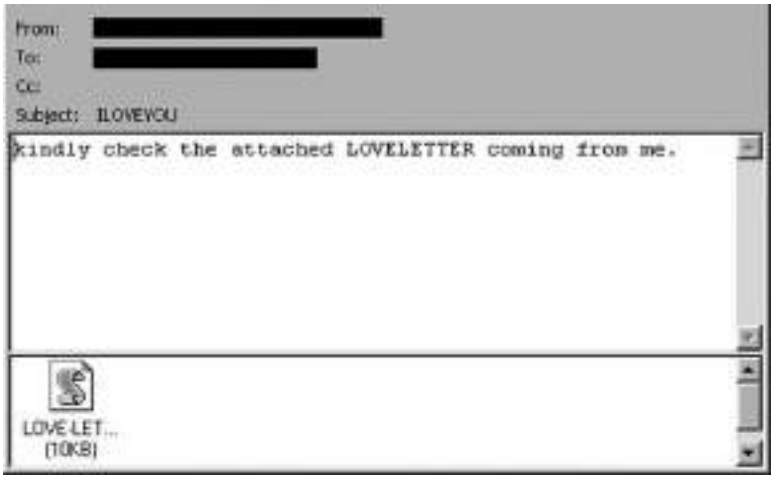


Figure 4-2: An example of the email sent by the ILOVEYOU worm

Historically, hackers have used worms as bragging rights. The more systems your worm was able to infect, the more credibility you got in the hacking community. Some worms, like Melissa, spread without a payload. Melissa was a macro created in 1999 by David Smith. It infected email systems using an attachment masquerading as a Word document. Clicking the file ran macro code that opened pornographic sites and sent copies of itself to everyone in the person's address book. Just because a worm doesn't have a payload doesn't mean it isn't harmful. Melissa took many email servers offline when the worm flooded them with emails until they crashed. Other worms, such as Code Red and SQL Slammer, also caused major outages of network services because of the amount of traffic they generated. Code Red appeared in July 2001 and within days infected nearly 400,000 systems

across the internet. SQL Slammer was even quicker: when it appeared in 2003, it infected some 75,000 hosts within 10 minutes. When worms do carry payloads, they usually include malware like ransomware or remote access backdoors (which we'll discuss shortly).

Trojans

Like the classic Greek story this type of malware is named after, a trojan is malware that pretends to be legitimate software while secretly running malicious tasks in the background. The trojan imitates many different software elements, including games, Word documents or PDFs, and even add-ons or macros. Once the trojan is installed, it usually begins running unwanted code, but it might not activate its full payload until a determined parameter is met or a command is sent. For example, many trojans send an HTTP request to a server controlled by a black hat waiting to receive commands from an infected computer. The fact that trojans mimic legitimate programs makes them very difficult to detect, which allows them to remain in place for long periods of time.

One of the most common types of trojan is called the *Remote Access Trojan (RAT)*. Its main purpose is to place a silent, undetected program on your system that allows an adversary to control your computer remotely. This program connects to a *command and control (C&C)* server, which allows an attacker to send commands to your computer without detection using normal traffic filtering procedures.

Essentially, the RAT uses normal traffic, such as requests for a website on the internet, to ask for additional commands from the C&C server. The attacker can then respond with additional commands, or in the case of the RAT, use the program to gain *backdoor access* to the system. (Backdoor access simply means access through unknown, unauthorized means.) This allows the black hat to use the system either to move to other targets or to attack other systems.

Ransomware

Ransomware is malware that uses encryption to lock a computer until a ransom is paid. Once the ransomware deploys, it encrypts specific targets, such as files in use, entire hard drives, or even entire databases, making them impossible to use. Because the files are encrypted, the computer can't read them, making them inaccessible until the black hat provides the key that decodes the content. The attacker holds on to the key until a ransom is paid, usually using a form of untraceable money like cryptocurrency.

Adversaries use this type of malware often because it has several strengths. First, it's easy to deploy and scale. A single infection can spread across an entire network or encrypt key critical systems, effectively shutting down an organization. Second, it's nearly impossible to bypass once deployed. Encryption is very hard to break, and many modern encryption protocols would take literally billions of years to crack. The only effective method to avoid paying the ransom is to maintain backups. But many organizations pay even when they have backups because of the time it takes

to restore the files. Third, it's cheap and effective. It costs nothing for an adversary to attempt infection, but a single success nets them thousands—or even hundreds of thousands—of dollars. More organizations opt to pay as the urgency of their services rises and cyber insurance that covers such attacks becomes more prevalent.

Spyware and Adware

Spyware and adware are probably some of the most annoying malware types that affect systems. Spyware steals data from your system, whereas adware injects ads into your system while you use it. Both kinds of malware typically infect web browsers or other programs that use the internet. They particularly like to hide inside add-ons or macros that users install in a browser. This allows them to track your browsing history, links clicked, and accounts accessed, all while serving up ads in annoying pop-ups that appear to randomly flash onscreen. Although these kinds of malware usually have less severe consequences than other types, they can cause your system to slow down and sometimes steal valuable personal data, including passwords. They can also lead to further infection from other malware kits because the loaded ads point to other types of malware, like trojans.

Rootkits and Bootkits

Rootkits and bootkits provide an attacker with unprecedented access to a system. A *rootkit* is malware that attempts to access a computer's internal system files, which are files that run the operating system. For example, a rootkit might replace the files that control logins. By doing this, a black hat can create a secret login that gives them full administrative privileges to the system while hiding the actual account from other users, so it remains undetected. Typically, these files are off limits to unauthorized users. To access these files, the rootkit takes advantage of a vulnerability that lets it run as an administrator. From there, it can make all sorts of modifications to system files, including adding users, changing file permissions, or changing a system's network settings.

A *bootkit* accesses and modifies a system's boot record, which is a file that starts an operating system when you turn on your computer. The boot record initializes many different configurations and usually loads additional software, like *hardware drivers*, which an operating system uses to interact with the computer's hardware, such as its keyboard and mouse. By modifying the boot record, the malware can change how the system functions, giving full access to a black hat or loading other malicious software, like bots, into the system (more on bots in Chapter 6). The other advantage to a bootkit is that many of the security functions that come with a system don't launch until after the system boots. This means that the malware can run without being detected by an antivirus engine or other security tool. Internet of Things (IoT) devices, which are small devices with network connections, such as thermostats or security cameras, are especially susceptible to bootkits, because they often run all their functions almost entirely off the boot record (we'll discuss IoT devices in Chapter 6).

Bootkits and rootkits are extremely difficult to detect on a system. Rootkits modify the programs that are designed to detect them. For example, they can modify your antivirus program so it skips the location at which the rootkit is installed when running a scan. Bootkits evade detection because traditional antivirus software only works once the operating system is loaded, which occurs after the boot record runs. This makes it very difficult to find anything wrong with the system until it's too late.

The typical way to detect both types of infections is through traditional symptoms, such as a slow-running system, missing or corrupted files, or weird running processes. You might also be able to detect a rootkit by using antivirus software installed on a USB flash drive or other media. Some systems also have the ability to do a *secure boot*. This modifies the boot process to detect errors or anomalies associated with bootkits to stop them before they can be run. Windows systems perform a secure boot by checking whether the boot record came from Microsoft. If the system discovers a modified boot record, the check will cause the system to stop booting. But even then, finding the malware is only part of the battle. Removing it is tricky, and often, it's better to completely wipe the system and reinstall it rather than risking letting the infection linger in some inaccessible corner.

Polymorphic Malware

The strongest and most dangerous form of malware is polymorphic malware. This malicious software has advanced capabilities that allow it to change its code based on certain factors, such as the kind of system it's currently infecting or the applications running on the system. This lets it adapt to the environment instead of just running a set payload, making it extremely difficult to detect by traditional means, because it might not be harmful until a specific situation is triggered. Once the trigger is set off, the malware activates, changes its code to run destructive operations, and begins to carry out its designed task.

Fortunately for us, polymorphic malware is extremely rare and crafted for very specific targets. Because of the time and resources required to create this type of malware, only state actors normally use it. One famous example of polymorphic malware is the *Stuxnet virus*. This malware was designed by the United States and its allies to infect Iranian nuclear centrifuges and stop them from functioning properly. The Stuxnet creators specifically crafted it to hide until it entered the centrifuge system, at which point it rewrote its code and infected the device. Stuxnet was able to remain undetected for many months, successfully sabotaging the program.

As technology advances, writing advanced pieces of code like polymorphic malware becomes easier. You might find polymorphic features integrated into traditional malware to provide it with added functionality. For example, by using machine learning theory and algorithms, it's possible to train malicious software to evade antivirus detection, even if an antivirus program detected that same destructive software previously. According to recent research from Hyrum Anderson, an attacker could run the malware through a set of antivirus engines using a machine

learning program, which would tweak the software's code slightly based on the result of the scans. It would repeat this cycle hundreds of thousands of times, tweaking the malware code until none of the engines returned a positive detection. The result will be a piece of harmful software that essentially runs the same as it did before but attracts no attention from antivirus engines. The adversary doesn't have to do anything to achieve this; the machine learning program does all the work for them.

How Black Hats Deploy Malware

Black hats deploy malware for a variety of reasons. To understand how and why an attacker releases malware on a system, let's walk through some typical attacks. Although all malware is different, most of it is installed in your system in consistent ways.

The first step to any malware deployment is to create the destructive software. Generally, adversaries do this in two ways: by taking advantage of an existing vulnerability or starting from scratch. Many black hats use malware that is already designed to exploit a vulnerability for their own attacks. This means they add specific payloads to an existing piece of malware based on their needs. A great example of this is the EternalBlue exploit, developed by the National Security Agency (NSA) and later used by attackers in several malware samples. The exploit allowed for remote code execution on a target Windows machine by taking advantage of the way the Server Message Block protocol, which manages network shares, handles certain types of information sent to it. In each case, the attackers added their own code to execute.

Writing new code from scratch is more difficult than using a framework, but it tends to be more effective. The reason is that the target system's protections won't have encountered the malware that was created before it's deployed. Because antivirus software relies heavily on code from actual malware samples, a brand-new piece of malware has a much higher chance of avoiding detection (as we'll discuss later in this chapter).

After creating the destructive code, the attacker moves to the next step: initial infection. They can install the damaging software on a system through a variety of means, but the most effective is social engineering, a process described in Chapter 3. Using phishing techniques, it's often easy to get users to download the malware and execute it. For instance, attackers like to hide unsafe code inside nondescript files types, such as Word documents or Excel spreadsheets. Both kinds of files allow you to create macros, where black hats can store malicious code that activates when users open the document, causing the initial infection. Using links to execute scripts that download harmful code to the computer is also common. Because a lot of malware needs human interaction to work, the more benign-looking the file or link, the better. This is especially true for trojans, which must remain on the system long term to be effective.

The initial infection will release the full payload, but that doesn't mean the malware has necessarily finished working. At this stage, some malware focuses on a specific action; for example, ransomware encrypts files or other

storage media. Other malware focuses on creating an APT, which is sophisticated malware that remains hidden on a network for a long period of time, gathering data and other information before executing a large attack. A RAT is an example of a possible APT; it allows an adversary to return to a system repeatedly through the backdoor to gather more information about its environment. APTs are extremely dangerous and difficult to deal with, because by design they avoid many traditional modes of detection. Even worse, some malicious software performs flashy attacks, such as encrypting all your Word documents, to hide the secret code that installed a rootkit onto the system. This is why digital forensics is critical in a malware attack (see "How to Defend Against Malware" next).

Once the infection is in place and the payload is deployed, the contagion can be spread. This might involve sending the malware in an email using the host system's contact list; moving through the network using transport protocols, such as File Transfer Protocol (FTP) or Hypertext Transfer Protocol (HTTP); or hiding in a file until a new user clicks it. While worms are particularly adept at spreading, ransomware and viruses can spread quickly and easily as well, although they might require user input to do so.

How to Defend Against Malware

The best way to defend against malware is to use anti-malware software, commonly referred to as *antivirus programs* (although they now guard against nearly all forms of malware, not just viruses). Antivirus programs are available from many commercial vendors. Microsoft systems also have a built-in antivirus program called Microsoft Defender (formerly Windows Defender). But merely downloading an antivirus program doesn't necessarily provide complete protection. To ensure that your system is safe, you need to get the proper software to diminish the risks your computer might encounter.

Antivirus software comes in two basic detection forms: signature and heuristic. The former uses a code signature to recognize malware. A code signature is a unique part of a destructive program's code that helps identify it. For example, let's say you receive a file in your email. A *signature antivirus program* scans the file and notices that the file includes the code *sparklekitten.exe* as part of a macro. It then compares that code to a database of known malware signatures and determines that *sparklekitten.exe* is flagged as malicious. The antivirus software will then alert you and possibly quarantine the file, depending on its settings.

Signature antivirus software is extremely fast because all it does is compare a piece of code to a signature database to verify whether it is malware. Figure 4-3 shows an example of such a program running on a Linux system. It also doesn't require many system resources, so it can run on most systems—even those with low memory capacities or a slow central processing unit (CPU), which is where the computer processes the instructions sent to it. However, for the software to detect the malware, its database must contain the malware's signature. This means that newly created malicious

software can avoid antivirus detection, at least until enough infections have been reported to the software manufacturers for them to add an accurate signature to the database. It also means that black hats can tweak the signature of a piece of malware to avoid detection (or use machine learning to change the malware, as discussed earlier in this chapter). Due to these pros and cons, it's best to deploy signature antivirus software on endpoint systems, such as desktops, mobile phones, laptops, and other systems that work on data, to avoid slowing down actual work being done on the system.

Figure 4-3: A screenshot of antivirus software running on a Linux-based system (this image was modified from the original created by SourceFire and is licensed under the GNU General Public License)

Heuristic antivirus detection adapts its detection based on the flow of traffic on a network, looking for abnormalities that are outside the normal traffic flow. What is normal will vary based on how the network is used, so the heuristic antivirus program has to spend time learning this baseline. Then it can notice anomalies. For example, if a RAT is installed on your computer, its first step is to send a message across the network to its C&C server. A heuristic antivirus program could detect this behavior and recognize that this isn't normal traffic (maybe because it occurs at an odd hour or comes from a system that doesn't usually send this type of traffic). However, as stated earlier, it's still possible for the RAT to imitate what the heuristic system considers normal.

Heuristic antivirus detection is very effective because it can detect malware that is brand new, as well as malware that is trying to hide, such as rootkits or bootkits. But it requires more setup and maintenance than signature-based detection. To find traffic that isn't normal, the heuristic engine first needs to know what normal looks like for your system. This means it must determine an accurate baseline before it can work

effectively, and that baseline must routinely update as the behavior of the system's users changes. Typically, you'll find heuristic systems in a few key high-traffic areas, such as on the firewalls that scan traffic coming into a network from the outside.

Antivirus software of either type scans most kinds of data coming into or leaving your system, depending on the settings or the specific product you're using. This includes documents like PDFs or pictures, applications like Excel or games, and even web traffic. The robust scanning toolkits in modern systems make it difficult for malware to find its way onto your screen by traditional means. However, good detection relies on your having a properly maintained antivirus program. You need to routinely update your software and perform automatic, periodic scanning. You should also make sure that your software is set up to scan all types of data, especially attachments in emails or downloaded files from the internet.

Despite the numerous technological advancements in antivirus software, it's still possible for attackers to circumvent those scans by understanding and avoiding the way they detect malware files. For example, you might receive a normal, clean Word document that the antivirus software won't react to; the document wouldn't show up as known malware in any signature database or appear to be unusual traffic. But when you open it, you might trigger a link that reaches out to the internet and downloads ransomware. Before your antivirus software has a chance to react, the ransomware might execute and lock you out of all your files.

You can also use file integrity tools to ensure that a file hasn't been modified to hide a trojan or other malware. A *file integrity tool* uses a file hash, which we'll discuss in Chapter 9, to check whether a file was modified. Most companies provide a hash of their applications or files on their websites. You can compare this hash to the hash of the file or application you downloaded to make sure a black hat hasn't added malware to it. If the hashes match, the file or application hasn't been modified.

To truly keep your computer safe from malware, you must practice your social engineering defenses. Weird links, unprompted attachments sent in email, and other such suspicious requests all indicate that an adversary might be trying to get you to download malware.

Exercise: Analyzing Malware and Managing Antivirus Settings

Anticipating the sources of malware and avoiding them are critical steps to keeping your system secure. After all, if the malware never gets onto your computer, you don't have to worry about it infecting your system. In this exercise, you'll use some free online tools to scan a PDF and find out whether it's infected with malware. You'll also learn more about the antivirus settings built into your computer, so you'll know what to do if you do accidentally download malware onto your system. By the time you finish the exercise, you'll know how to identify and defend against all sorts of malware threats.

Analyzing Malware in Attachments

Let's say you've received a strange PDF that looks like it came from a friend. This friend has sent you emails before with attachments like this, but you weren't expecting anything from them and aren't sure whether or not it's malicious. One solution is to ask your friend if they sent the file; another is to just delete it and move on with your life. However, say your friend isn't available, and you really want to know whether or not the file is safe. Well then, you'll need to do some malware analysis.

To complete this exercise, you can use a PDF created for this book, called *maliciouspdf.pdf*, which is available at *https://nostarch.com/cybersecurityreallyworks/*. Alternatively, you can analyze any file you'd like; just make sure to label it *DO NOT OPEN* so you don't forget it's potentially malware.

First, you'll need to verify the type of file that was sent. Office documents, executables, media files, and PDFs aren't usually harmful unless you open them. But some, such as *.js*, *.sh*, or *.script* files, could execute upon downloading; others, like *.dll* files, might be activated by other processes once downloaded. The best option is to download a file using a virtual machine. A virtual machine is an environment that's isolated from the rest of the physical computer, so if it does become infected, it very likely won't infect the entire system. For malware to break out of the virtual machine, it would need to be very sophisticated software.

Because not everyone has access to a virtual machine, another solution is to download the suspected harmful file to a cloud platform. For example, if you use OneDrive for Windows, you can save the file directly to your online OneDrive folder instead of to your computer. Often, this will trigger an antivirus scan on the file, which might indicate it's infected without you having to do further analysis. If you don't have an online folder, the next best option is to use a flash drive or external hard drive. Although it's still possible that malware saved to external storage will infect your main system, it reduces some of the risk. Just be sure no critical files are on the drive before you save the suspicious file.

When you download the file, make sure you *do not open it*. Click the **Save As** option and move it to a folder called something like *Do Not Open* or *Malware* to remind you and others not to accidentally open it. Also, make sure your system doesn't autorun any files by default. Many systems present you with an **Open With** option when you download a file. *Do not* select this option.

Once you've safely saved the file, you can begin running it through the analysis tools. The first website you'll use is VirusTotal (*https://www.virustotal.com/*), which you used in Chapter 3 to analyze a suspicious URL link. This time, you'll use the File Analysis feature. When you load the page, you'll see an option to upload a file for analysis. Click **Choose File**, navigate to your saved file, and select it for upload. Figure 4-4 shows an example using the *maliciouspdf.pdf* file.

Figure 4-4: The malicious PDF ready for upload

Once the file is uploaded, click **Confirm Upload** to begin the analysis. VirusTotal runs the file through multiple antivirus software packages and returns the results, informing you whether or not the file is malicious. Keep in mind that VirusTotal will specify the number of engines that found malware in the file. If even one engine reports the file as being unsafe, you can assume it's malware regardless of whether the others indicate it's clean. Figure 4-5 provides an example of the output for *maliciouspdf.pdf*.

Figure 4-5: The results from VirusTotal

As you can see, 37 different malware scanning services reported that the file contained malware. This file was created using a well-known exploit whose signature would have been loaded into many antivirus programs, making it easy to detect. But this might not always be the case. A file might

contain new malware that doesn't yet have a standard signature, causing VirusTotal to report the file as clean when it's actually infected. To deal with this possibility, you'll need to use another familiar tool, Joe Sandbox.

As you learned in Chapter 3, Joe Sandbox (*https://www.joesandbox.com/*) allows you to analyze a link or file using a cloud platform that functions like an actual system. In this exercise, you'll use it to open the file in a safe environment where you can thoroughly analyze it. To start, upload the file, as shown in Figure 4-6.

Figure 4-6: Uploading the file on Joe Sandbox

Once the file is uploaded, the sandbox will take a few minutes to finish the analysis. Joe Sandbox opens the file and then runs it through several different antivirus scans. Figure 4-7 shows that these antivirus scanners identified *maliciouspdf.pdf* as malicious.

Figure 4-7: Joe Sandbox results for maliciouspdf.pdf

Joe Sandbox also provides additional analysis aimed at understanding what type of malware might be embedded in the file and how the malware would have affected your system if downloaded and opened. This can be especially helpful if you think you might have accidentally opened the file and need to figure out what sort of malicious activity it performed. Figure 4-8 shows some useful parts of the report on *maliciouspdf.pdf*.

This analysis gives you a detection level, in this case, malicious, and also provides two ways of classifying the file: Signatures and a Classification map. Recall that many antivirus programs use signature files to detect malware.

The Signatures section displays the signatures that match the file being analyzed. The Classification section then provides an estimate of what type of malware the file contains based on the scans and signatures detected.

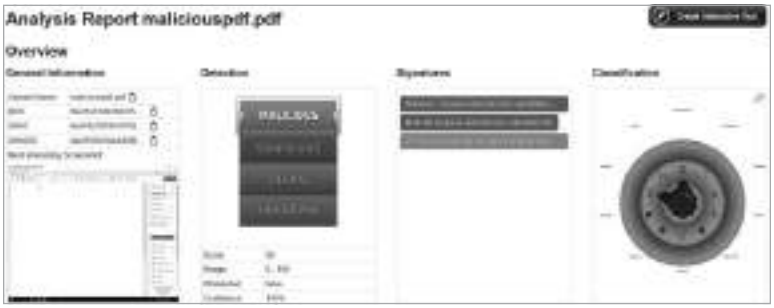


Figure 4-8: Overview of the analysis for maliciouspdf.pdf

Next, Figure 4-9 displays a breakdown of what processes were created when the file was executed in the sandbox.

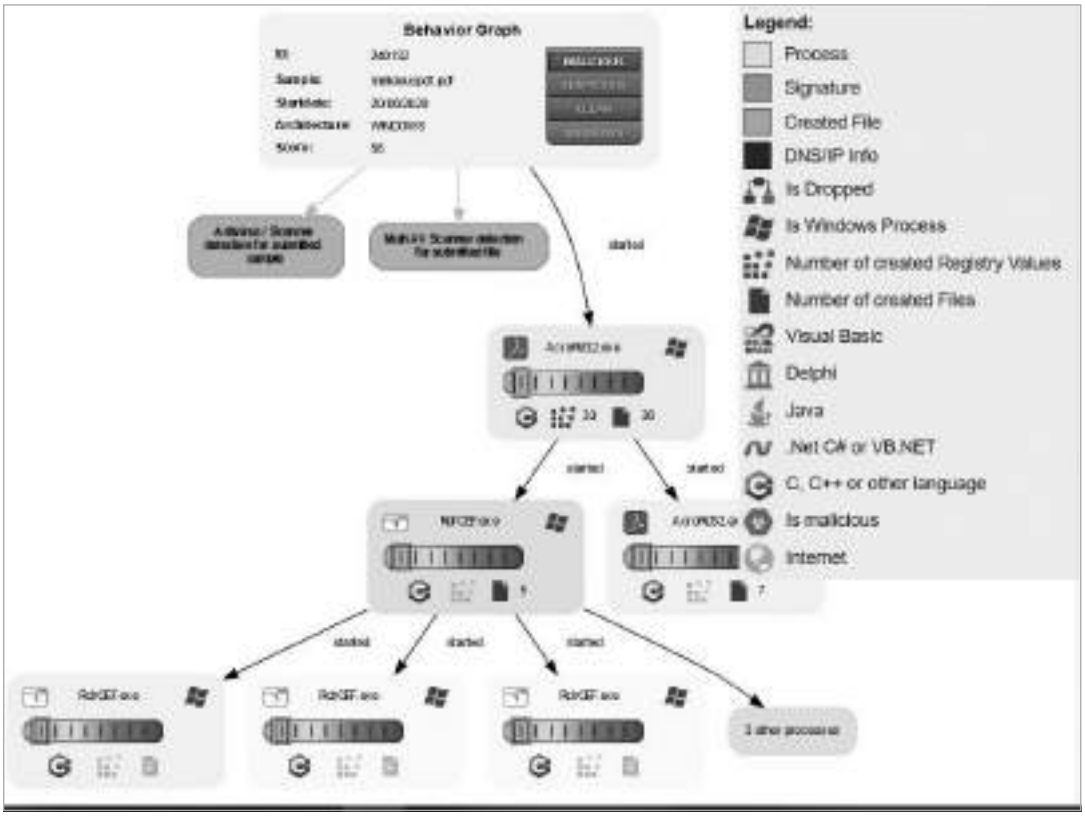


Figure 4-9: Processes created by maliciouspdf.pdf *when it's opened*

The report also lists any child processes created by parent processes. In this example, the Adobe Reader process started when the PDF was opened and then spawned two child processes. One of these processes spawned even more child processes. This part of the report can be useful to identify any processes that aren't normally found in this type of file execution, such as the child processes here, which aren't associated with Adobe.

Reviewing Antivirus Settings

Now that you've analyzed the suspicious document and know it's malware, you need to review the antivirus settings on your system to make sure your computer is protected against infection.

Built-in antivirus software doesn't come with macOS, but you can use several third-party options to keep your system safe. All Windows 10 systems come with Microsoft Defender installed, which is integrated into the operating system's security settings and provides a wide range of malware protection, including against ransomware. Although other commercial products, such as Sophos, Check Point, Avast, and Symantec, offer additional features, we'll focus on Microsoft Defender because it's free and built into Windows 10 by default.

macOS

Most modern antivirus software comes with an Apple-compatible version. If you're looking for a free program, Avast and Sophos provide effective products. But when you use the free versions, the features are limited. Both products offer paid versions that provide more robust features.

Although not as many existing types of malware are programmed to attack Apple computers as Windows systems, it's still essential to scan your computer regularly. It's also critical to set automatic updates so you'll have the latest signatures as soon as they're available. Be wary of where you get applications, even from the official Apple store. Malware, especially ransomware and trojans, is often disguised as applications, and some of these have been known to slip past Apple's verification process and been placed in the official store.

Windows 10

To access the Microsoft Defender settings, locate the security dashboard by entering **security** into the search bar located in the taskbar at the bottom-left corner of the screen. Click **Windows Security** and then click **Virus & Threat Protection** to open the Microsoft Defender settings for antivirus protection, as shown in Figure 4-10.

You can perform a few different actions using these options. First, you can run a manual scan using the Quick Scan Option. This will scan the most likely places a virus or other malware might be located, and it does so in no time at all. As you can see, the last scan done on this system took one minute and 26 seconds to scan 42,363 files. Quick scans are beneficial if you think your computer has been infected with a well-known, ordinary piece of

malware. But not all malware hides in the obvious files. To make sure your system is truly clean, it's best to scan more than those files that a quick scan inspects. To do this, click **Scan Options** below the Quick Scan button.

Figure 4-10: Virus & threat protection settings

Microsoft Defender offers several options for scanning other than a quick scan. The *full scan* option inspects your entire operating system. This is a more intense scan than a quick scan, so it can slow down your computer and takes a long time to complete. But it provides a comprehensive report of your system by checking every nook and cranny. You can also do a *custom scan* of specifically chosen files. It's best to use this option when you know exactly what type of malware you're dealing with and know where it likes to hide. For example, you could use the analysis report from Joe Sandbox to determine which files or folders a malicious file interacted with, and then scan them to see whether the malware infected your system as well. The *Windows Defender Offline scan* is essentially a boot sector scanning option. It restarts your computer and scans it before the system has a chance to fully boot to ensure that rootkits or bootkits are unable to hide or modify processes once the system is started. Figure 4-11 shows these scanning options.

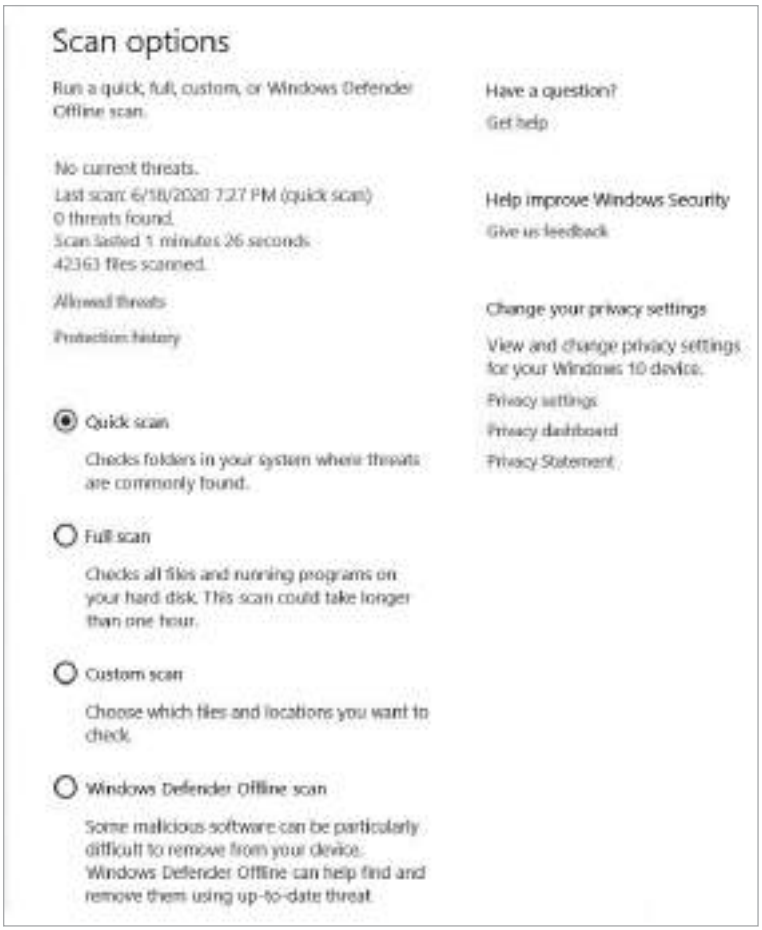


Figure 4-11: Microsoft Defender scan options

When you return to the Virus & Threat Protection dialog, find the Protection History option. This shows you a list of all the threats that Microsoft Defender has detected and what it did to neutralize them (Figure 4-12). If you're ever concerned that a virus wasn't mitigated properly or that the antivirus software might have removed legitimate files accidentally (which is rare but happens from time to time), you can use this dialog to view the recent activity. Figure 4-12 shows an example of two threats that were discovered and quarantined on the system (in case you were wondering what happens when you add malicious PDFs to your system as part of this exercise).

The additional options below Virus & Threat Protection Settings determine how Microsoft Defender runs, including whether it offers

real-time protection, which stops malware from installing, or whether it submits samples of malicious code to an antivirus database to be used for signatures. Microsoft Defender does provide some cloud-powered heuristic resources as well, but overall, it's a signature-based system. By default, it will automatically update its list of signatures, but you should make sure these are up-to-date nonetheless. You can manually check for updates by clicking Check for Updates in the Virus & Threat Protection dialog.

Figure 4-12: Microsoft Defender protection history

The final setting you'll want to check is *ransomware protection*, as shown in Figure 4-13. To guard against ransomware, Microsoft Defender controls access to folders and provides backups using certain cloud platforms. In these settings, you can see which files are protected and modify some of the protection settings. Keep in mind that Microsoft Defender isn't foolproof. If you maintain regular backups on an external hard drive or cloud system separate from your main storage, you'll further help protect your system should it be locked by ransomware.

You've now learned how to analyze a file for potentially embedded malware and have prepared your Windows system to reduce the threat of infection. Both skills are imperative for defending your system from any malware threat. Combining these skills with those you learned in Chapter 3 will make it even more difficult for a black hat to compromise your system. Prevention is one of the best ways to defeat malware. Keep in mind that if malware never makes it onto your system, you don't have to worry about what it does.

Figure 4-13: Microsoft Defender ransomware protection settings

Conclusion

This chapter focused on the various types of malware and their distinct characteristics. Viruses and worms, the most traditional malware types, typically carry payloads that perform malicious actions when activated. In addition, worms are designed to spread the malware to new systems. Ransomware encrypts files and holds them for ransom, allowing black hats to extort money from their victims. Trojans hide in plain sight, allowing backdoor access or even installing rootkits or bootkits, which infect a system's heavier security areas to cause modifications to its operation. Spyware and adware are sometimes more annoying than they are malicious, stealing traffic and serving up ads, such as pop-ups that appear out of nowhere. But the worst type is polymorphic malware. This advanced software can change its code on the fly, allowing for advanced persistent infections that are incredibly hard to detect.

In this chapter, we also explored how to defend against malware. The key is to combine social engineering defense skills with advanced anti-malware software. By being mindful of where and what you click and what you download, you can prevent an infection. If a system does get infected, antivirus software helps detect the malicious code by using either signature or heuristic characteristics. Signature detection is fast and requires few resources but isn't as advanced as heuristic detection, which compares traffic to a baseline to determine whether it's normal or unusual. By combining these tactics, you can keep your system free of infection and running the way it should.

5

PASSWORD THEFTS AND OTHER ACCOUNT ACCESS TRICKS

We use access control procedures to ensure that only authorized people can access a system, open a file, or run a piece of software. It's part of an organization's everyday management; every person who works for a business deals with it in some form or fashion, whether by logging into an email account or sharing a file with a client. To better manage such a sizable and critical topic, the security world often divides access control into three categories: authentication, authorization, and accounting.

In this chapter, we'll explore how various methods of authentication and authorization can keep your systems safe, and how we use accounting to keep track of everything done in a system. Throughout the chapter, you'll also learn how black hats circumvent access control.

Authentication

Authentication is verifying that someone is who they say they are. Let's say a knight comes to your castle's gate. He could be an enemy knight or a friendly knight, depending on his shield's crest. To authenticate the knight, you send a squire to check the knight's crest. If he has a friendly crest, you let him in. If he has an enemy crest, you keep the gate closed.

Cybersecurity professionals meticulously distinguish authentication from a related concept, *identification*. You might use some form of identification to state who you're trying to authenticate as, whereas authentication proves you're actually that person. For example, when you enter a username and password, the username identifies you. But the username alone doesn't prove you're truly that user. By entering the password, you validate that you're the user associated with that ID.

Types of Authentication

You can authenticate a person or system in many ways: using a password, using DNA, or even by knowing the way a person speaks. Each of these authentication methods has its own strengths and drawbacks. To better classify different methods, cybersecurity specialists group them into the following five types.

Type 1: Something You Know

Type 1, something you know, is typically a piece of information you've memorized. The most widely used version is the password, because passwords are incredibly easy to set up and maintain, and a vast array of systems can use them. Consider how many times a day you use passwords to access accounts or systems.

But passwords aren't the only form of Type 1 authentication. Another common one is the security question, also known as the *cognitive password*. Typically, you'd use one of these to reset a regular password when you forget it. The answer to the question the system asks is a response that only you would know, such as your mother's maiden name or the street you grew up on. However, with the spread of social media, it's become easier for black hats to discover the answers to these questions and use them to reset your password.

In fact, Type 1 is the easiest form of authentication to break, because it's not necessarily unique or complex. Think of the passwords you use, and answer these questions honestly. How many passwords do you have that are:

- Only used in one place?
- At least 12 characters long?
- Made up of lowercase and uppercase letters, numbers, and symbols?

If any of your passwords fail to meet the preceding requirements, they're susceptible to attack. Adversaries use a few clever techniques to try to break passwords, including brute force. To brute-force a password, a computer system automatically runs through every combination of possible password

characters, and then attempts to use them to log in to a system. Often, this process takes mere hours to accomplish a guessing task that a person would waste months doing manually. Although this technique might seem time-consuming, given enough time brute-force attacks always work, even if it takes years to go through every possible combination. The shorter the password, the easier it is to crack. In addition, you never know when an attacker might get lucky on the first try.

Another attack adversaries use is the *dictionary attack*. This attack relies on common words or combinations to shorten the amount of time it takes to brute-force a password. For example, one of the top passwords, year after year, is *qwerty* (the letters on the top row of a standard US keyboard). A dictionary attack might run through all the passwords stored in a file, including *qwerty* and some other common ones, narrowing down the combinations the attacker must try. The less complex a password is, the easier it is to break using a dictionary attack.

That said, black hats typically don't have to brute-force a password or use a dictionary attack. It's much easier to get the user to tell them the password. Using social engineering, or by simply looking for the password on a sticky note at a person's desk, attackers constantly get people to reveal their passwords. Once they have it, there is nothing to stop attackers from using it. Type 1 authentication doesn't use additional checks to confirm the person entering the password is who they claim to be; it merely reads the password, and if it's correct, lets them in. So although Type 1 authentication might be cheap and easy to deploy, it's not the most secure, which is why we rely on other types.

Type 2: Something You Have

Something you have is an entity, whether it's a physical object or a digital artifact on a computer, that you must present to authenticate to the system. The object, which is unique to you, usually provides a code or key to the system. There are many ways to implement Type 2 authentication; examples include smart cards, key generators, and digital certificates, which you'll learn more about in Chapter 9.

Type 2 authentication is stronger than Type 1, because the attacker needs to steal the object to bypass authentication, which is much harder to do than guessing a password. But Type 2 is also more complex and expensive to implement, which is why it's not as common. The reason is that Type 2 requires special equipment and additional hardware. For example, if you added a key card reader system to log in to your computer, not only would you have to purchase, install, and maintain the reader, but you'd have to purchase key cards for every user, as well as keep additional cards on hand for when someone inevitably loses one. However, over the last several years, new ways of implementing Type 2 authentication have made it much cheaper to use. The most familiar of these is the text verification code.

Systems that use text verification codes usually prompt you to enter a phone number when you first set up an account. To access your account later, the system asks you for permission to send a verification code to your

phone. Once you agree, the code is sent by text message (or phone call, although this isn't as common). You then enter this code in a field on the system or app. This process proves that you're the owner of the account, because theoretically, only you have access to that phone number.

One of the most secure methods of Type 2 authentication is the smart card. In a system that uses smart cards, each person receives a unique card. On the card is a chip that contains an encryption key (we'll discuss these keys in Chapter 9). When you slide the card through a chip reader, the reader checks the key and authenticates you. No card? No key? No login. The use of an encryption key instead of a code makes it an even stronger authentication, because it's nearly impossible to guess the key. This method has become more common over the years: chips have been used on credit cards since the mid-1990s but didn't become ubiquitous in the United States until 2015. Figure 5-1 shows an example of a *Common Access Card (CAC)*, a type of smart card the US federal government and military use to access desktops and buildings.

Figure 5-1: A Navy Common Access Card

A Type 2 authentication object doesn't have to be physical. It can also be a digital entity, such as a *digital certificate*: a piece of data stored on a computer that identifies and authenticates it when it tries to access another system. For example, when a system queries a database for information, that system might send the database a digital certificate to prove it's authorized to make the query. The certificate is stored on the hardware, sometimes in a special chip known as a *Trusted Platform Module (TPM)*. There are many layers of security between the TPM and the other hardware on the system, making it extremely difficult for a black hat to access the certificate without having physical access to the machine. We'll discuss certificates and encryption in more detail in Chapter 9.

Type 3: Something You Are

Although Type 1 and Type 2 authentication provide a lot of protection, they both have the same drawback: neither can be uniquely tied to a single person, making it possible for someone else to use their credentials. This is where Type 3 authentication excels. Type 3 systems use a person's unique biometric signature as the authentication method.

A *biometric* is some physical or behavioral attribute of a person. The most common example of a biometric used for authentication is a fingerprint, but others include retina scans, facial recognition, voice, or even DNA. A biometric might also be a behavior, such as the way a person walks, or their signature. In theory, these attributes are universally unique, meaning no other person in the world will have the same attribute as another. Table 5-1 lists some commonly used biometrics.

Table 5-1: Types of Biometrics

Biometric type	Scanning data
Fingerprint scanner	The swirls on the pads of fingers
Hand scanner	Unique patterns on fingers and palm
Iris scanner	Shape of eyeball
Retina scanner	Pattern of blood vessels at back of eyeball
Face scanner	Pattern and shape of face

To set up a biometric system, the people using it must provide their biometric signatures, which are stored in a database. For example, if your workplace wanted to use fingerprint scanners on all doors, it would have to scan your fingertips. When you used the scanner at the door, the signature gathered would be compared to the signature stored in the database. If they're the same, you'd be allowed in.

This can be a tricky process, because every scanner has a different sensitivity level. For example, if the back door's scanner is too sensitive, it might deny you entry, even though it should have allowed you access via that door. The rate at which this happens is known as the *false rejection rate (FRR)*. On the other side of the scale, if the scanner's sensitivity is too low, it might grant access to someone who isn't in the database. The rate at which this occurs is known as the *false acceptance rate (FAR)*. People who design these systems must figure out how to set the scanner in a way that minimizes both the FRR and the FAR. We call this sweet spot the *crossover error rate (CER)*. The need to find the CER makes installing a biometric system a bit of a trial-and-error process. Also, the type of scanner will affect the system's success, because simpler scanners don't create as complete or complex a signature as more sophisticated scanners.

Type 3 is the strongest form of authentication. Despite what movies would have you believe, it's extremely difficult to impersonate a biometric signature, especially if you're using a high-quality scanner. Some of the scanners currently in use can even detect a heartbeat in a finger to determine whether it's connected to a living person. This means that to get the signature, the correct person must use the scanner. This severely reduces an adversary's ability to break into the system or steal credentials, the way they might with Type 1 or Type 2 authentication.

In addition to a high-quality scanner, these systems need large databases in which to store the signatures. These requirements make biometrics considerably more expensive to implement. Even though scanners have become

cheaper, the ones placed into mobile devices don't provide the same accuracy as those used as stand-alone systems. This leads to the second issue with biometrics: bad scanning. Because a signature is extremely unique, any change to it can lead to an FRR error. For example, if you burn the finger you use for a fingerprint scanner, the scanner might no longer work. Even doing something like shaving a beard can confuse facial scanners.

Biometrics can be difficult to use and adjust, so it's important to have a backup authentication system should the biometric system fail. For example, on a mobile phone with a fingerprint or facial scanner, the authentication system usually provides an option to enter a pin number if the scanner doesn't recognize the person. This way, you can adjust the biometric scanner's sensitivity over time without fear that it will lock you out of the system you're trying to access.

Types 4 and 5: Something You Do and Somewhere You Are

The other two types of authentication typically supplement one of the other forms of authentication rather than acting as stand-alone methods. They serve as part of multi-factor authentication (discussed in more detail next).

Something you do (Type 4) is an action you must take to authenticate. For example, to get into a mad scientist's secret lab, you might have to pull a certain book on a bookcase to reveal the entrance. Only the people who know which book to pull will be able to authenticate. Another example of Type 4 is the *CAPTCHA*, which is a test that proves you're a human, not a script designed to automatically log in to an account. By choosing the right pictures, such as images of cars from a group of photos, you prove that you're a real person. The Type 4 authentication method doesn't provide adequate protection on its own, however, because anyone can perform the action, as long as they know what needs to be done. (In the mad scientist's case, an adversary could simply watch through a cracked door or over someone's shoulder to see what choices a person makes to authenticate.)

Somewhere you are (Type 5) is the location of a person when an authentication is made. If the person isn't in the right location, the authentication fails. On its own, Type 5 falls short in the same way as Type 4; there's just no way to verify that the individual in the location is the right person without using other authentication methods. However, when paired with different authentication types, Type 5 provides additional protection from black hats trying to access an account. For example, if you register your account as being in Little Rock, Arkansas, and someone tries to log in to the account from Hong Kong, the system can recognize that it's probably not you and send an alert. Although it's possible to spoof your location on the internet (more on that topic in Chapter 6), this protection adds another layer of defense that the attacker must bypass to access the account. The more layers in the way, the harder it is for an adversary to succeed in breaking into the system.

Multi-Factor Authentication

Using more than one type of authentication is called *multi-factor authentication*. It can help make up for the weaknesses of any one type of

authentication. Recall the problems with Type 1 authentication. Something you know is guessable, can be written down or transferred to other people, and anyone can use it regardless of whether they created it. But if we add Type 2 authentication to a system that already uses Type 1, we fix many of those problems. Because they must have some additional item to authenticate, such as a verification code from a text message, an attacker can't just guess the password to get in.

You must have multi-factor authentication if you want to protect your system from the modern techniques black hats use. The most common multi-factor authentication strategy today is adding a phone number or email address to an account and having a verification code sent to it when you attempt to log in. You're then prompted to enter this code after you enter your password. This might seem like an unnecessary nuisance to get into your social media account, but as discussed previously, attackers can obtain passwords in many ways. Having multi-factor authentication makes this more difficult for adversaries, and many times, they'll give up once they run into obstacles. At the very least, it provides extra time for you to react to the adversary's actions before they break into the account.

Adding extra defenses is a security strategy known as *defense in depth*. Basically, the more barriers you put up, the harder it is for an attacker to easily gain access. This not only helps supplement weaknesses found in individual defenses (like the weaknesses in guessable passwords, for example), but also gives white hats time to become aware of, learn about, and respond to the black hat activity. As discussed in Chapter 1, awareness is poison to attackers; the more layers you construct, the more likely it is that white hats will be alerted to the nefarious activity and stop it.

Authorization

Once users authenticate, they must be *authorized* to do certain actions. The friendly knight from the earlier example doesn't necessarily get into the tower where the princess lives just because he's friendly. Instead, he might be authorized to enter the castle but not the tower. As with authentication, authorization is based on who you are in an organization. A friendly knight and the king might be able to enter the castle, but only the king, princess, and princess's attendants might be authorized to enter the princess's tower.

Authorization is vitally important. To use a more modern example, if your company has two engineers authorized to make changes to the web server, it can be difficult to track who is actually making changes. When an attacker breaks in or a malicious action occurs, finding the source could be more difficult. If only one engineer is authorized to make changes to the web server, you know exactly who is responsible for any changes or even any malicious actions that might happen.

A *security kernel* program, which is part of an operating system, typically enforces authorization for changes made on that system. How the security kernel enforces authorization depends on the type of access control scheme in use. There are five common access control schemes: mandatory access

control (MAC), rule-based and role-based access control (both commonly abbreviated as RBAC, though I'll use the full names to avoid confusion), attribute-based access control (ABAC), and discretionary access control (DAC). Let's examine each in turn.

Mandatory Access Control

MAC is a centralized system in which a central authority strictly enforces access control. MAC provides a high level of control over who can access files, systems, or software, but it doesn't allow a lot of flexibility. The system tests all attempts to access a resource against a central security policy to determine whether to grant access. If the type of access requested doesn't exactly fit into the policy, it's denied. A single administrator or group of administrators controls the policy.

To create such a policy, the administrators might use labels to determine what sort of authorization a person should need to access each resource. For example, the military uses a system of classifications with three base categories: confidential, secret, and top secret. Any new file gets assigned a classification based on the policy. If a document is given top-secret classification, only people who are authorized at the top-secret clearance level can access it, and a person with secret clearance can't override the system to view top-secret documents.

Rule-Based Access Control

Rule-based access control uses specific rules to determine what type of authorization to grant. This is an incredibly rigid system. Variations in context matter little, and if no rule matches the access request, most systems use an *implicit deny*, meaning they automatically reject the action. This helps prevent the rule-based access control policy from becoming too bloated. It's much easier to define what you want to happen in a system rather than trying to keep track of everything you don't want to happen. That said, this system forces the administrator to map every possible authorized action, or people will likely get denied. In certain contexts, such as complex environments with many requirements, it might not be feasible to use a rule-based access control system. Unlike MAC, a rule-based scheme must separately set rules on every asset.

A great example of rule-based access control is file permissions. Most operating systems grant access to files based on rules that take into account who is accessing the file and what actions they're authorized to do. Each file has its own set of these rules, which the system reads to determine whether to grant access. For example, a system administrator might be able to read and write (meaning *change*) a system file, whereas a standard user would only be able to read it.

Role-Based Access Control

Role-based access control uses a user's role to determine their access to the system. Unlike with rule-based access control, the roles apply across the

entire system, not to individual objects like files. For example, if you work in human resources, you might be given the role of Human Resources Officer. This means that when you log in to your computer, you can access the human resources department's shared folder and log in to the employee record database.

Role-based access control provides a lot more flexibility than either MAC or rule-based access control. The system administrator can create new roles as needed to provide the necessary level of access. This method also makes it much easier to manage the access of a large group of people. For example, every person working in a call center might need access to a set of resources, such as a list of customer accounts. By creating the role Call Center Employee, the system administrator can easily give every call center employee the authorization to do their job without having to go into the system and individually grant access to every resource.

The main drawback of role-based access control is that it often leads to *privilege creep*. Privilege creep occurs when an individual or group gradually gets more authorizations, until they have permission to do more than what their job requires. This becomes a problem when the controls can no longer keep an account from doing something it's not actually authorized to do. For example, you might take over a role in your company temporarily while a new person is hired. If the authorization granted by that role remains once the new person joins, you might have access to resources you shouldn't.

The roles can also be too broad. To guarantee that a large subset of people fit a role, each role might have a wide range of access. For example, the administrator role often receives a full set of permissions, whether or not that user needs all administrator functions. Or a human resources employee might not necessarily need to see every document pertaining to an employee. They might only be a recruiter, so they don't need access to current employee files.

Such broad access roles make it easier for a black hat to gain access to files, accounts, or systems. To counteract privilege creep, security professionals usually apply the concepts of least privilege and separation of duties. Someone who has *least privilege* has only the privileges necessary to do their job. For example, a graphic designer might be able to update pictures on a website but not log in to the administrator side and change the website name. A task with *separation of duties* requires multiple people to finish it. For instance, two people might be needed to create a check and pay a vendor: someone to draft the check and another to sign it once they've verified it's a valid payment.

Attribute-Based Access Control

ABAC is similar in flexibility to role-based access control but helps alleviate the drawbacks of privilege creep. Essentially, ABAC uses descriptors (aptly named attributes) to determine what sort of access a person or system needs. You can think of each attribute as its own mini-role. When a person or system tries to access a resource, the system reviews their attributes to authorize access.

This scheme solves the human resources problem mentioned earlier. With an ABAC system, a member of this department would receive the attribute *human resources* on their account. However, if they're also a recruiter, they would have the attribute *recruiter* as well. The human resources attribute allows them to read general human resource documents, but because they also have the recruiter attribute, they can't read the files of current employees.

Attributes can be combined in many ways, allowing you to fine-tune access control while still maintaining the broadness and flexibility of role-based access control. Privilege creep remains a possibility; however, it's much easier to protect against it with ABAC, because it's possible to severely limit the access of attributes without making the system cumbersome, like a strict MAC system.

Discretionary Access Control

DAC is the most flexible and least secure of all the access control systems. Under DAC, whoever owns the object, whether it's a file, application, or system, decides who has access to it. This creates plenty of flexibility, because the owner can grant and deny access as necessary. But this system is also insecure, because no central authority decides how access is granted or denied, making it more likely for someone to receive unauthorized access.

A great example of DAC is in document services, such as OneDrive or Google Drive. When you create a document in one of these services, it's placed in your personal account. You can then grant access to that document by sharing it with a third party, like a co-worker who is working on the same project. No system tells you who can or can't receive the document. It's up to you to decide who you share it with, and once they no longer need access, you can remove them from the share.

This scheme can lead to issues; for example, if you enter the wrong email address or share the file with an employee in another department who doesn't need to see the document. Therefore, we typically only use DAC for very limited applications, like sharing documents, rather than as a full access control system like MAC.

Accounting

Accounting is making sure that every action taken on a system or network generates a record. Although accounting doesn't protect an account or system from direct access by an attacker, it's still vitally important to maintaining an organization's security. If you're unable to verify the activity happening in an account or system at any given time, you won't be able to know whether you've maintained security. In addition, if an incident does happen, it can be difficult to find the attack details and remove the adversary from the environment. Therefore, it's important to maintain accounting using a twofold process: enabling logging and performing routine auditing.

Logging

Logging is a catchall term for capturing events that happen on a system during its operation. Each system has its own logging method. For example, if you're logging an application, the logs would include records of exactly what the application code did when the application ran. If you're logging an account, it would include a timeline of when the account signed in, from where, and what was accessed. Most logging follows similar conventions when it processes events. This includes the level of severity, what information is gathered, and where that log is stored.

Events in logs are usually broken down into levels to indicate an issue's severity. The standard ranking method is known as *Syslog*, which starts at 7 as the least severe and ends with 0 as the most severe. Table 5-2 lists each level and its associated severity.

Table 5-2: Log Severity Levels

Value	Severity	Description
0	Emergency	System is unusable.
1	Alert	Action must be taken immediately.
2	Critical	Critical conditions.
3	Error	Error conditions.
4	Warning	Warning conditions.
5	Notice	Normal but requires special handling.
6	Informational	Informational messages.
7	Debug	Debugging messages.

The most important events, at level 0, indicate that system hardware had a failure that caused it to break down. Level 1 indicates a failure that is causing the system to run improperly, for example, because it crashed. Level 2 is used for failures that are causing operations within the system to malfunction, such as an application crash, despite the system continuing to run. Level 3 includes conditions that cause an error but don't necessarily disrupt operations; for example, trying to access a document that doesn't exist anymore might result in an error, but that error probably won't crash the computer. Levels 4 and 5 provide information on events that might need an additional look but aren't generally considered security risks. An example might be a user failing to log in to their account once in an hour. Level 6 events provide information about what the system did: for example, if it opened a file or made an authorized connection. We use level 7 only in rare cases when we're trying to find a problem with a system's underlying operations. Usually, level 7 isn't turned on by default, and you should only use it with extreme caution. Logging generates many entries, and it's possible to fill up log storage with level 7 events alone.

Typically, you should log as much information as possible while still maintaining the information's usefulness. If you log too little, you'll miss

events that might help you understand an incident. If you log too much, you'll be drowning in data that might conceal key information you need to find quickly. Many logging agents allow you to filter out events to make the logs more digestible. You could filter out events at level 6, such as successful logins, to make other events, like a failed login attempt, easier to find. But you must carefully tune this filtering to avoid missing malicious events. (I discuss what those malicious events might look like later in the chapter.)

Additionally, you must decide how long to keep the logs. Organizations normally make this decision by considering the type of log, how much information the log contains, how much storage is available, if there are any legal or compliance requirements, and other factors. The primary concern is whether you'll be able to determine what happened if you need to go back to a certain event or date. For example, if you find out that an employee's computer was breached, you'll need to access all the activity on your network from that computer on that day to determine what the black hat did.

Most organizations used to keep logs for 90 days, but many have now transitioned to yearly logs as storage becomes less expensive. They usually store their logs on large servers with solid-state hard drives to best protect them against problems that might take the system offline or erase its data. Whichever media you choose, you should store it offsite. That way, the logs are more likely to survive a disaster, such as a ransomware infection, at the main site. Cloud services also offer the ability to stores logs for a customer.

No person should have write access—the ability to change a log—or the ability to remove a log from the system. The entire point of logging is that it's an accurate account of every event that's happened on a system or network. If a person can modify those events, the log becomes less trustworthy. Making the logs uneditable helps eliminate inside threats.

Auditing

Just as important as capturing logging information is looking through the information. We *audit* not only to find malicious activity, but also for routine maintenance. For auditing to work, we can't do it only when we have a problem. We need to catch problems as soon as they happen, before a hardware failure, a software bug, or a black hat compromise does significant damage to our systems.

As mentioned earlier, every event on a system should produce a log. Deciding how many of those logs to audit can be tricky. If you audit too much, you'll waste time slogging through normal events. If you audit too little, you'll miss key indicators of malicious activity. Instead, you'll need to prioritize your organization's critical assets. For example, it's not reasonable to audit every workstation in your organization, but you could prioritize critical servers on your network to check every day.

Even if you narrow down the critical assets, there are still likely thousands of events that you might have to review. To help narrow those down even more, the next step in auditing is to set up alerts about important events. This is where Syslog can come in handy. By using the various levels of severity in the Syslog standard, you can have high-level alerts sent

directly to you. Events such as remote logins, password changes, and account lockouts could also generate alerts.

But setting alerts might still not be enough to capture malicious activities in your organization, because attackers have become adept at hiding. Instead of brazenly attacking a network or system, adversaries will combine different small attacks that culminate in a bigger compromise. They're also incredibly patient, slowly trying different attacks before finally finding a way in. For example, instead of brute-forcing a login by trying several passwords in quick succession, adversaries might try one password every six hours. Once they find the correct password, they might wait weeks or even months before they log in to the account. Then they'll sit and wait, watching internal network traffic or activities on the system as they gather information to use in a final big attack that infects a database with ransomware.

This type of attack methodology is difficult to detect through alerting alone, because none of the activity, up until the attack on the database, is out of the ordinary. Even though there might be indicators of oddity—for example, if the black hat logs in from an unusual location—these might not be enough to trigger an alert, especially if the strange events occur across multiple devices.

To better track this type of malicious behavior, you can use a *security information and event management (SIEM)* system. A SIEM system correlates logs from across all the devices and networks in your organization. This means that every event captured in your organization is fed into the SIEM system for processing. The SIEM system then weeds out the normal events, or any false positives, and provides an audit log of every suspicious event. You can also program it to generate alerts when critical events occur, like a brute-force attack on an account. The best part is that because the SIEM system gets logs from multiple devices, it can correlate activities to identify suspicious behavior. For example, a remote login to a workstation isn't necessarily suspicious. But a change to the firewall to allow remote connections to a workstation, followed by a remote connection to that workstation, might raise red flags. The SIEM system would make special note of that activity in its logs. If that workstation then connects to a database as an administrator and begins downloading all the files, this series of unusual events could warrant a full-blown alert.

SIEM systems are very robust, but they require upkeep. At their core, they still rely on rules and other indicators to determine whether an event, or set of events, is malicious. As organizations evolve, they must update these rules to ensure they remain in line with the current state of their systems and network. For example, if you add a new server, you'll need to add its logs to the SIEM system and create rules based on what services that server provides. If you fail to maintain your SIEM system, you'll likely miss critical events that you could use to find a compromise.

Indicators of Attack

Indicators of Attack (IoAs) are events that point to malicious activity taking place on a network, device, or account. This activity could be the result of

malware, a black hat, or an insider threat. Table 5-3 describes several common IoAs, some examples of each, and what they might indicate. This list is not exhaustive by any means, but it should give you a good idea of what you need to consider when setting up logging, auditing, or SIEM rules.

Table 5-3: Indicators of Attack

IoA	Example	Possible activity
Unusual outbound traffic	Device connecting to known malicious IP address; device using unusual protocols, like FTP; large amount of queries to a particular website or group of websites	Malware contacting a command-and-control server; files being removed; backdoor being accessed
Internal device running network scans	Workstation or server sending out ping packets	Malware or black hat looking for other systems to compromise
Account login from location outside business area	Logins from foreign countries; logins from multiple different locations at the same time	Compromise of account credentials by black hat or botnet
Changes to system settings	Firewall changes or port changes to allow new traffic to connect, such as opening FTP ports; new accounts added to the system; an account given administrator access; new automated tasks created	Compromise of system by malware or black hat
Changes to email settings	New inbox rules created; new mail flow rules; dramatic increase in email activity from an account	Compromise of email account; use of email to send out spam or phishing attacks
Application or system making irregular connections	System in an external network connecting to an internal system; application making new or unusual requests, such as trying to download data from a read-only database; system attempting to access a device it's not authorized to or that is outside its normal workflow (for example, a front desk workstation trying to connect to an HR database)	Compromise of application or system by malware or black hat; attacker then uses compromise to attempt to steal data from, or gain access to, other systems on the network
Multiple rapid failures	Several failed login attempts; multiple access request failures; multiple system failures	Black hat is attempting to access a system or account, for example, using a brute-force attack on an account login; might be trying to use system failures to bypass normal security controls
Unauthorized programs or processes running	Programs are set to run on startup and aren't part of normal business software; processes consume a lot of memory or CPU resources	Malware, particularly a trojan
Activity outside of normal operation time	Website queries, emails sent, applications run, or logins made during non-business hours	Compromise of system by malware or black hat, including possible backdoor or trojan

Exercise: Setting Up Accounts in Windows 10 and macOS

The best way to understand how authentication and authorization systems can affect your use of a system is to manage the accounts on your home computer. Whether you use a Windows or Apple system, you should be able to create accounts and then control the access they have to certain parts of the system. In this exercise, you'll configure the security settings for your account on either a Windows or Apple computer. You'll then create a new account and give it access to a shared folder. Although simple, this exercise uses all aspects of the authentication and authorization principles discussed in this chapter and will give you a real-world sense of how you can protect these systems against unwanted access.

Windows 10

Windows comes with a variety of built-in security features that you can use to protect your account from unauthorized access. Many of these features are already on by default. But it's a good idea to check them to make sure you're getting the most protection for your system. To do this, enter **security** in the search bar in the lower-left corner of your screen. The task menu appears, containing a list of several applications. Click **Windows Security** to open the security settings. Figure 5-2 shows the Windows Security dialog that should open.

Figure 5-2: The Windows Security dialog

As discussed in Chapter 4, this is where you can access many of the security features that a Windows system includes. This time, you'll click **Account Protection**. Figure 5-3 shows the resulting dialog.

In this dialog, you can see the settings that relate to account sign-ins. At the top, you should see the name of the account that you're currently signed in with. Just below that is the *Windows Hello* sign-in option. Windows Hello provides a biometric login by using your facial structure to unlock your system. (Note that this feature is only available on compatible systems

that have a web camera.) Although biometric logins are stronger than your traditional password, it's important to note that this feature has had problems recognizing faces and allowing other, similar-shaped faces to log in.

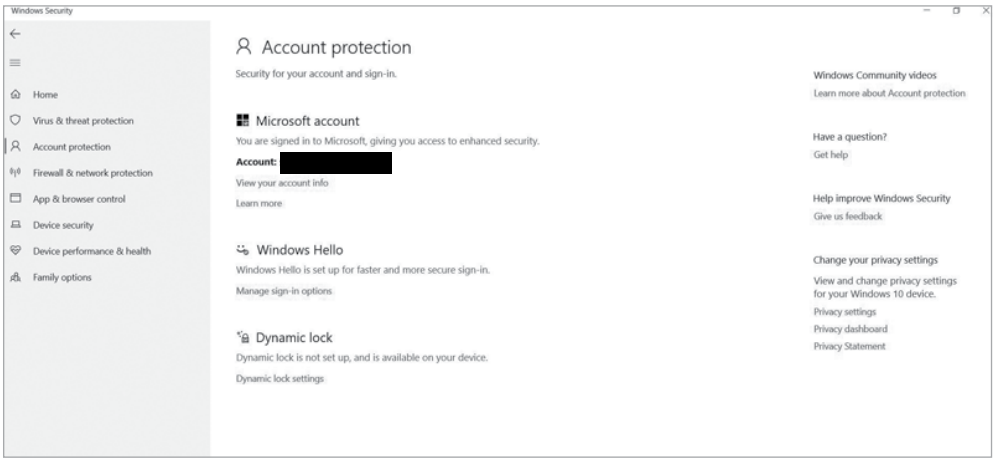


Figure 5-3: The Account Protection dialog

Just below Windows Hello is the *Dynamic Lock* option. This allows your system to pair with another device, such as a laptop, tablet, or phone via Bluetooth, and automatically lock the computer's screen when it loses its connection to that device. Dynamic lock can offer peace of mind whenever you have to leave your system unattended, but it might become frustrating if you move around a lot and have to keep unlocking your screen. Also, even though Bluetooth is a short-range radio technology, it can connect over surprisingly long distances. For example, leaving your office to go to the break room for a cup of coffee might not be far enough to lock the computer, so it might remain open for any passerby to access.

Now that you're more familiar with some of the security features in Windows 10, let's look at the specific account settings you can change. In the Account Protection dialog, within the Windows Hello section, click **Manage Sign-in Options**. Figure 5-4 shows the dialog that should appear.

From this dialog, you can investigate and change how your system logs in to your account. The list of options is diverse enough to provide the level of security you need while also making it easy to use. Keep in mind that if a password or other authentication activity isn't easy for users to implement, they'll likely use it incorrectly or bypass it altogether.

The *Windows Hello Fingerprint* option is a biometric fingerprint scanner. It requires either a built-in or attached fingerprint scanning device to work (and as you can see, the computer in Figure 5-3 doesn't have this option available). The *Windows Hello PIN* option provides a PIN that you can use to log in as an alternative to a traditional password. It's meant to be faster to use; but remember, you still need to make sure you create a long, difficult-to-guess pin (at least six to eight numbers) to guard against a brute-force attack.

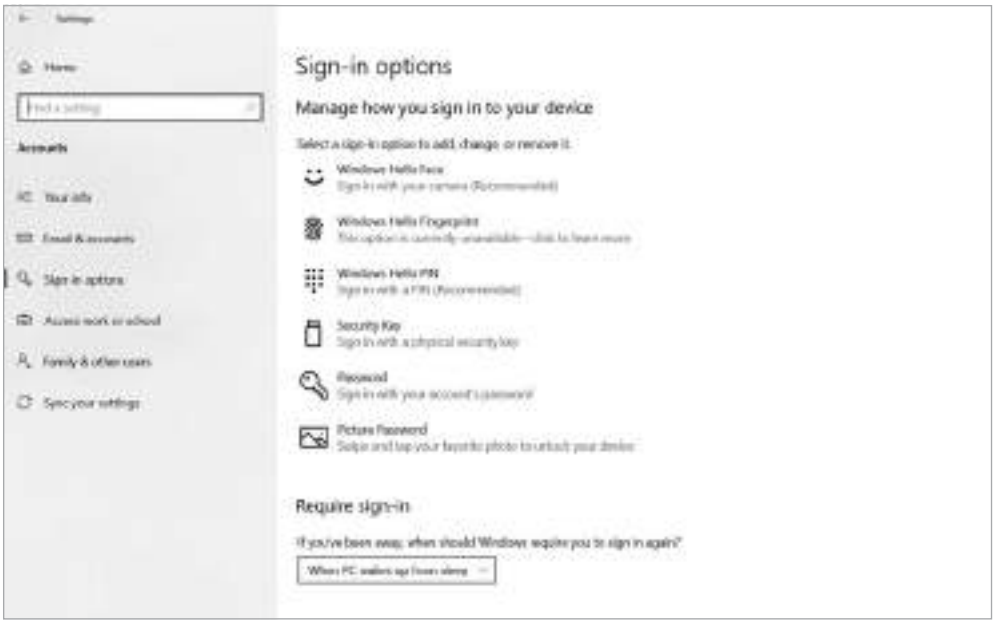


Figure 5-4: The Sign-in Options dialog

The *Security Key* option generates a token—a Type 2 authentication factor—whose unique key lets you log in to the system. You must have a physical security token to pair with this device. Typically you'll only find these keys used in businesses that can afford to buy them, but inexpensive commercial tokens, such as the Duo or Google Authenticator apps for phones, are also available.

Windows also offers two password options, a traditional password and a picture password. The picture password requires you to choose a picture and then draw on it. For example, you could choose a picture of a face and then draw circles around the eyes. To access the system, you'll need to replicate your actions when shown the picture at login. This is Type 4 authentication (something you do). It's usually considered the weakest form of authentication, because most people's gestures follow the natural paths of the picture. For example, if you chose a picture of a flag on a flagpole, you might draw a line down the flagpole, which is fairly easy for an attacker to guess, because it's a predictable action.

Take a moment to try each option to see which one you like the best. Just because you've always used a password doesn't mean it's right for you. Adding a different type of authentication might provide better security or functionality for your device.

Before moving on, let's look at one last setting. Below the different sign-in options is a drop-down menu for *Require Sign-In*. This lets you determine at which point your system should require you to log back in after being inactive for a time: the two options are *Never* and *When PC Wakes Up from Sleep*. You should always require a new login after the system wakes up. This keeps your

system safe from unauthorized access should you stop using it but forget to log out. This setting should be turned on by default.

Adding a New Account

Now that you've chosen your security settings, you can create a new account. To do this, click the **Family & Other Users** option in the sidebar on the left. Figure 5-5 shows the screen that appears.

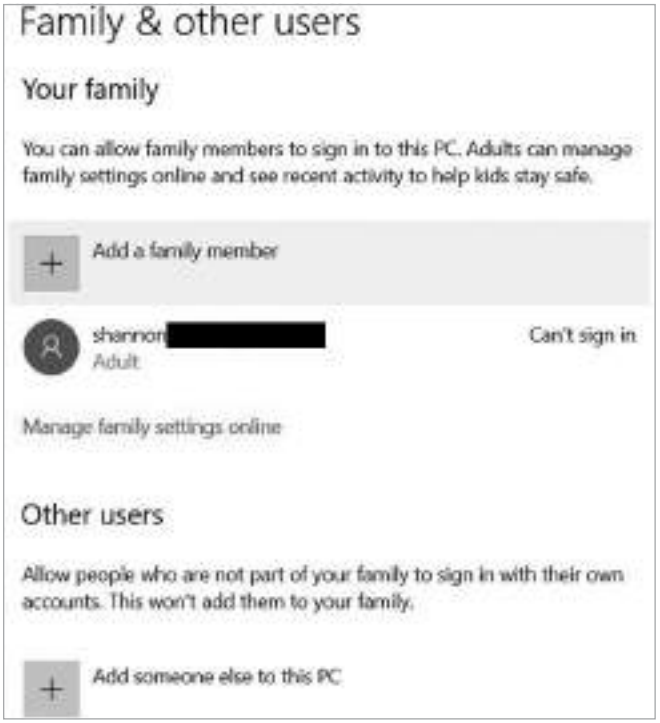


Figure 5-5: The Family & other users dialog

This dialog allows you to add other accounts to your system. At the top is the *Your Family* section. In the Windows 10 ecosystem, a family member is an account with additional auditing features and parental controls to help keep children safe. This feature also allows you to share apps and other purchases across accounts. But for this exercise, we'll focus on the second section, *Other Users*. This option lets you add another, traditional user account to your operating system.

Let's add a new account. Later in this exercise, you'll learn how to share a folder between the current account you're using and the new account you create. Click the plus icon next to *Add Someone Else To This PC*. A dialog opens asking how the new user should be able to sign in and for a corresponding Microsoft account. If the new user doesn't have a Microsoft account, click **I Don't Have This Person's Sign In Information**. Similarly, on the next page, click **Create Without a Microsoft Account**. Figure 5-6 shows the dialog that appears now.

Create a local account for the system. A *local account* is linked to only one computer rather than being part of a larger network. In Figure 5-6, I created a user called Sparkle Kitten, but you can name your new user whatever you want. Fill out the information and click **Next** when you're done.

Figure 5-6: Creating a new account

After creating the account, you need to decide whether or not to make it an Administrator account in the Change Account Type dialog (Figure 5-7).

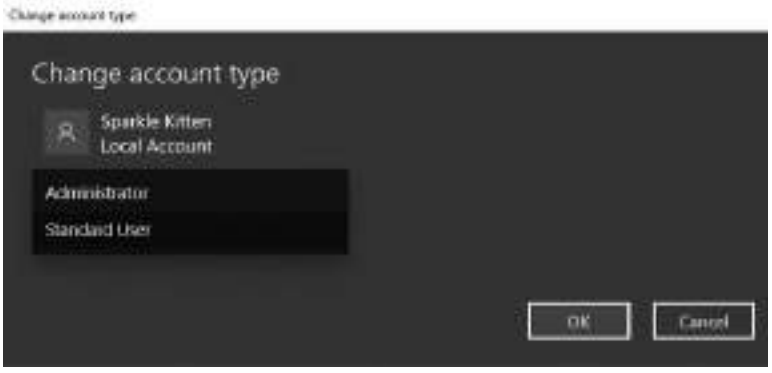


Figure 5-7: The Change Account Type dialog

Administrator accounts can access system files and make other changes to the system that could potentially be harmful, so determine whether the person using the account really needs to be an administrator

before assigning them that type of account. Remember to keep the principle of least privilege in mind when making this decision. For this exercise, a Standard User account will suffice.

Sharing a Folder

An important part of access control is being able to limit control based on the principles of least privilege and a need-to-know basis. Let's use those principles now by creating a folder and sharing it with the new account you just made. Create a new folder in the original account by right-clicking an empty area on the Desktop and clicking **New ▸ Folder**. It doesn't matter where you create the folder, but Documents or Desktop are probably the most convenient locations.

Once the folder is created, you'll need to access its properties to share it. Right-click the folder and then click **Properties**. Figure 5-8 shows the dialog that appears.

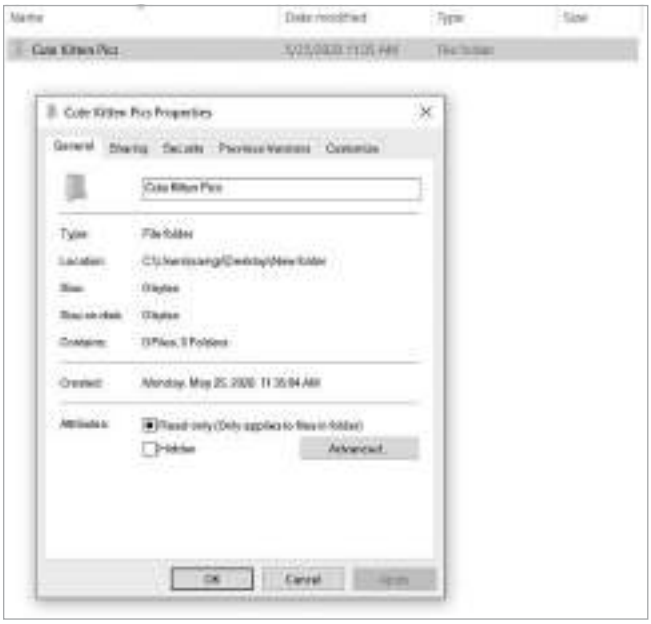


Figure 5-8: Folder properties

With the Properties dialog open, click the **Sharing** tab at the top. This tab contains settings you'll use to share the folder (Figure 5-9).

In this dialog, click **Advanced Sharing**, which brings up the dialog shown in Figure 5-10.

NOTE *Advanced Sharing might not be an option for you if your account doesn't have administrator privileges. If you can't access the Advanced Sharing dialog, click the **Share** button to share the folder with the user. A wizard will guide you through the sharing steps.*

Figure 5-9: The Sharing dialog

Figure 5-10: The Advanced Sharing dialog

Select the **Share This Folder** option, which opens a new option that asks you for a share name and displays a Permissions button. For now, click **Permissions** (Figure 5-11).

Figure 5-11: Sharing permissions

The Share Permissions dialog is similar to the ones you might see in many access control systems. For each group or user, it shows you a list of permissions corresponding to the folder. Although basic, these permissions provide a full spectrum of control. *Read* allows a user to see what's in the folder but not delete or rename the folder. *Change* allows a user to rename or remove the folder, and *Full Control* grants the user complete access to read, write, rename, or delete the folder.

To add a specific user, click the **Add** button to display the dialog shown in Figure 5-12.

From here, you can add other users to the folder's permission list, and then give them the appropriate permissions. Just enter the name of the user in the white box near the bottom of the dialog. Because I named my new user Sparkle Kitten, I entered that name. Next, click **Check Names**, and the system should automatically fill in the complete username, as shown in Figure 5-12. If the username doesn't fill in as shown in the figure, be sure to check your spelling. It must be precise to find the correct username.

Click **OK**, and the user you selected should appear in the list below Everyone in the Share Permissions dialog. You can then click that username and set its permissions. Generally, it's best to default to just Read access, so users can see what's in the folder but not make changes.

Figure 5-12: Adding a user

Let's also look at the Security tab next to the Sharing tab in the Properties dialog. The Security tab provides some of the same functionality as Sharing. Essentially, the Security tab shows every user or group with access to the file or folder and the permissions they have. You can use this tab much in the same way you used the Advanced Sharing feature: to add a user and give them permissions to the folder. It's important to note that any files or folders you place in the shared folder will need to have the same user permissions as the shared folder for the user to access them. The shared folder permissions only give users access to the folder, not necessarily to the contents within. Figure 5-13 shows an example of what the Security tab looks like.

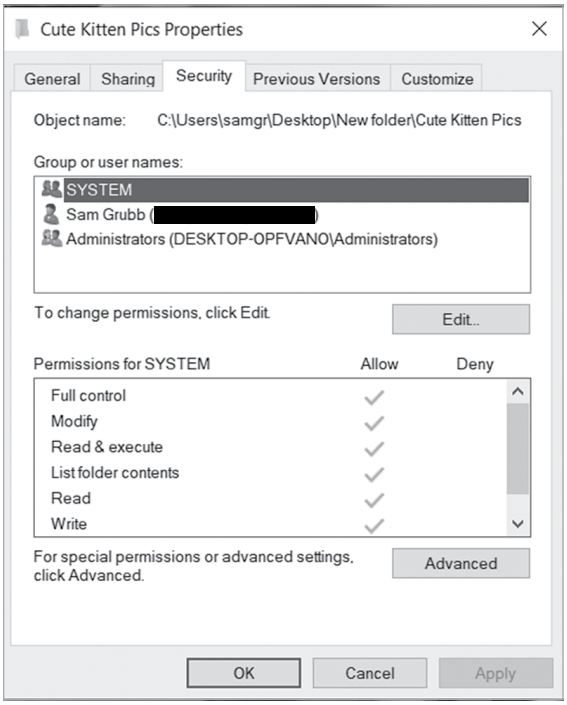


Figure 5-13: The Security tab

You now know how to configure the security settings on your account, add a user, and give that user permission to view a shared folder. Although this exercise might not seem that significant, we just covered nearly all of the access control principles, including authentication, authorization, least privilege, and even the DAC model. Now let's look at how to configure the security settings on macOS.

Access Control on macOS

macOS provides its own access control and authentication challenges. One challenge is that Apple systems streamline many of the controls that are normally accessible in a Window environment. This gives the user a limited amount of control over how to manage their systems unless they're prepared to do a deep dive into system files and edit them manually. This isn't recommended unless you have a lot of experience, because you can easily edit a system file incorrectly and break your system. In this section, I'll provide a general overview of the controls available to you and where you can locate them.

Account Management

Let's begin by looking at account management for macOS. This operating system provides some useful controls that you can use to further secure your system, specifically how and when the system automatically logs out for you. To start, click the Apple symbol in the top-left corner of your screen, and then click **System Preferences** to open the application (Figure 5-14).

Figure 5-14: The System Preferences application

This app is your one-stop shop for most settings and configurations for your system. Click **Security & Privacy** to display the dialog in Figure 5-15.

Figure 5-15: The Security & Privacy dialog

As you can see in the dialog, there aren't many options to choose from. You should set the timer that determines how long your system can be inactive before requiring a password to at least five minutes, if not lower, depending on your security needs. You can also access advanced settings, as shown in Figure 5-16, by clicking the lock icon in the bottom-left corner, entering your password, and then clicking the **Advanced** button in the bottom-right corner. (You'll need to have administrator permissions to do this.) The advanced settings allow you to add additional security to your system by requiring an administrator password to change any settings that would affect more than one user. You can also change the log-out timer for inactivity. This timer is different than the previous inactivity timer, because it fully logs you out of the system rather than simply requiring you to enter your password again. It's best to set this option to at most 30 minutes.

Once you've configured these settings, return to System Preferences. Then click **Users & Groups** in the last row of icons. A dialog should open showing all the users for the system. Once again, by clicking the lock icon in the bottom-left corner and entering your password, you can access some additional settings for each user. At the bottom of the Users & Groups dialog, click **Login Options** to display the dialog shown in Figure 5-17.

These settings mostly pertain to which options appear on the login menu, such as password hints or the ability to shut down or restart the computer without logging in first. You can also add new users to the system by clicking the plus icon just below Login Options in the bottom-left corner. Now let's look at how to share files between user accounts.

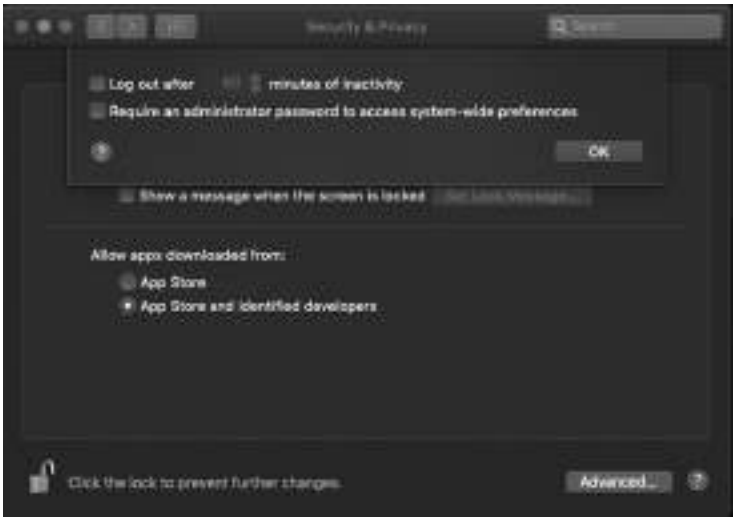


Figure 5-16: The advanced Security & Privacy settings

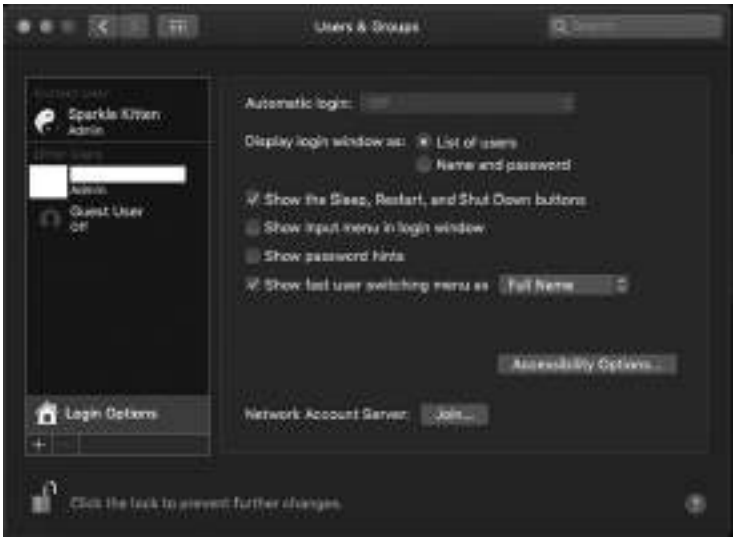


Figure 5-17: Login options

File Sharing

macOS streamlines file sharing; every system has a built-in file share associated with each user account. You can control and grant access to this file share from a central dialog. Return to System Preferences and click the **Sharing** icon to display its dialog, as shown in Figure 5-18.

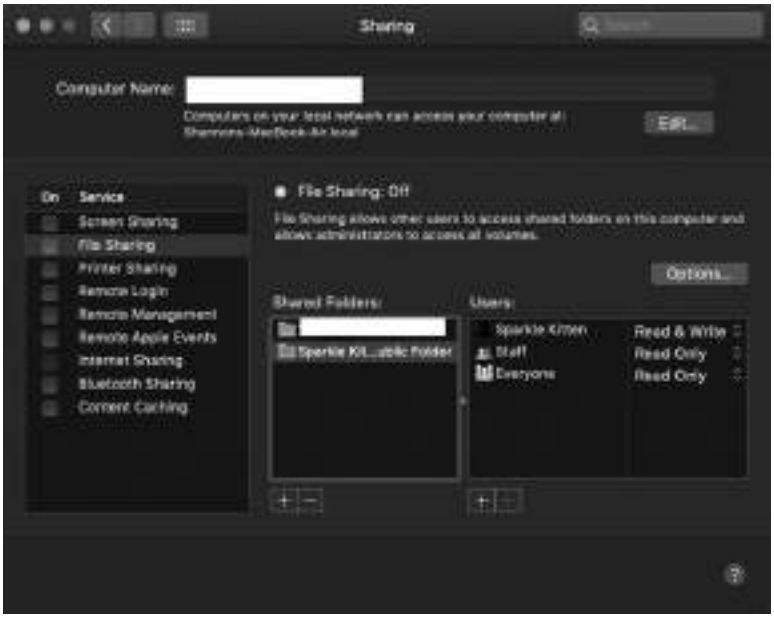


Figure 5-18: The Sharing dialog

From this dialog, you can control settings for all forms of sharing, including remote login, file sharing, screen sharing, and more. In the Sharing dialog, you can see a list of shared folders. By clicking the plus icon at the bottom of this list, you can add another folder you want to share. Clicking a shared folder in the list will also display options for who can access it, as well as what type of access they can have. There are fewer options here than on Windows, but they're extremely easy to understand. *Read* lets someone see the folder, whereas *Write* lets them add items to or remove items from the folder.

You've now learned the basics of account management, access control, and file sharing for Windows and macOS. Using this knowledge, you can better control who has access to your system and how you share information with others. Remember to only grant people the bare minimum of access they need for their given tasks. Giving someone full rights when they only need Read permissions is just asking for disaster.

Conclusion

In this chapter, you learned about the three parts of a robust access control system: authentication, authorization, and accounting. When authenticating users, make sure you're using a method that fits your environment's needs while still maintaining security. Multi-factor authentication can provide an extra layer of security, and this precaution can determine whether your accounts get compromised. When authorizing users to perform tasks on your systems, use an access control system that ensures people maintain the

least amount of privilege they need to do their job. You should also distribute duties between multiple individuals to guarantee that no one person has too much authority.

Auditing lets you maintain a good grasp of what is happening inside your organization. To audit effectively, set up logs, establish good practices, and watch out for various IoAs. With these processes in place, you'll create a reliable and efficient access control program in your organization. You'll let the right people in, keep the wrong people out, and ensure you've tracked every event.

6

NETWORK TAPPING

Today, nearly all devices are connected to a network. This allows them to communicate with other devices locally, such as when connecting to printers, and over the internet, such as when visiting a website or using an online application. Although these connections provide a lot of utility, they also allow black hats to find those devices. By exploiting the way networks run, attackers can see your traffic, pretend to be a legitimate device, or even determine how traffic moves on the network.

In this chapter, you'll learn more about how we create wired computer networks, as well as some details about the devices that make up a network. You'll learn how adversaries steal traffic on these networks and gain access to network devices. You'll also learn how to defend against network attacks using firewalls and intrusion detection systems (IDSs). We'll conclude this chapter by configuring your device's firewall.

This chapter focuses on wired networks only. In Chapter 8, we'll talk about wireless networks, which have their own set of challenges. Even if you use a wireless network, your traffic will almost certainly flow through a wired network eventually.

The Basics of Network Design

Networks allow two or more devices to communicate with each other, either wirelessly or using cables. You can imagine the link between devices as somewhat similar to the power lines connected to your home. The lines, some of them attached to poles, connect the power station to your house. The power station most likely is connected to another station, which is connected to another one, until it reaches the place where the power is being generated, such as a dam. In the same way, your computer is connected to other devices in a chain until it reaches another device on the other side of the network.

Networks are made up of more than just cables. They also include devices, such as routers and switches, that help move traffic and navigate the connections. Routers provide the main connections between two different networks. In Chapter 2, we discussed the difference between public and private networks. A router is the device that directs traffic between these two types of networks. A switch works within a network, directing traffic between the devices connected to it. Together, these two devices transport your traffic from one point to another. Figure 6-1 shows an example of a typical rack of routers and switches that a large enterprise network might use. Smaller networks, like a small business or home network, might have one switch or use a switch/router combo.

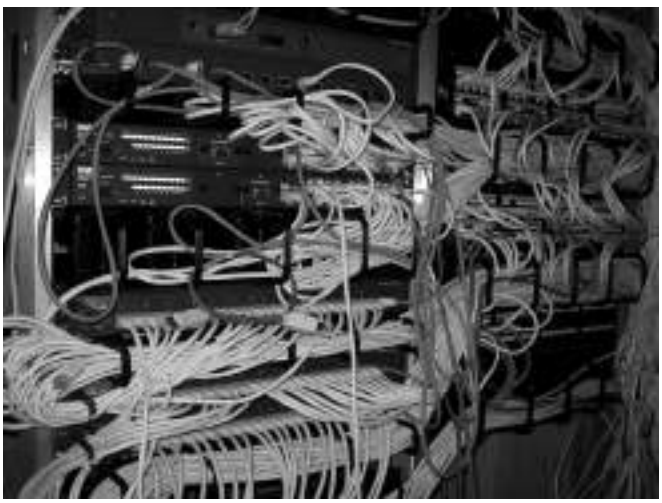


Figure 6-1: A typical network rack setup (image altered from the original created by Adrian Sampson under the Attribution 2.0 Generic [CC BY 2.0] license, https://creativecommons.org/licenses/by/2.0/)

Let's look at the traffic flow and connections when you're accessing a website. When you send a request to visit a website, you use the HTTP protocol. The protocol helps categorize the data, so devices understand how to interpret it. Most protocols are also given a *port number*, a special number assigned to the protocol so network devices recognize what type of data is being sent and how they should handle it. For HTTP, the port number is 80, or 443 if the data is encrypted. When a switch or router sees port number 80, it recognizes the packet as HTTP traffic and automatically knows how to process it without having to look at all the data inside the packet. This makes it quicker to send the packet on to its destination.

Using HTTP, you send out a request from your computer in the form of a *packet*. That packet includes the website request, the port number (80), your IP address, and the IP address of the website you're visiting. This request is sent to a switch. The switch looks at the request and determines whether the destination is on the current network. Most likely it's not, so the switch passes the packet to the router that connects your house to the internet. As mentioned in Chapter 2, this router is known as the default gateway.

Once the router receives the packet from the switch, it checks its list of all the other networks connected to it. There are many types of networks, but the most common are the *local area network (LAN)* and *wide area network (WAN)*. A LAN is a small network that connects devices in the same physical area. Examples of LANs include office buildings, homes, and even airplanes. WANs connect devices across a wide geographical area. The internet is made up of many, many WANs, all linked into one giant network. The router examines the destination IP address of the packet it receives from the switch and determines whether the address is on any of the LANs or WANs it's connected to. If the router doesn't know where that destination IP address is located, it usually has a default network where it sends traffic.

Regardless, the router passes traffic to another router, which follows the same process. This chain continues until the packet finally reaches a router that is connected to the LAN where the destination device is located. The router then sends the packet to the switch on the LAN, which finds the destination device. There can be more than one switch on a LAN, even hundreds of them on large networks, like those maintained by Google or Amazon. Switches pass traffic between them much like routers do, except they use the MAC address instead of just the IP address. Once the destination device receives the packet, it reads the data inside, such as the website request, and responds. The entire process just described then happens again in reverse: the original source of the request now becomes the destination of the response.

This is the basis of network communication. But it can get more complicated. Other devices read network traffic, too, and might even modify it before sending it on. For example, a special type of server, called a *proxy server*, takes a packet from one network connection and passes it to another connection. Often, the proxy server modifies the original packet in some way, such as by changing the destination IP address from a public address to a private address. Another type of proxy reads website requests and determines

whether they conform to what an organization has determined to be appropriate to view at work before sending them out to the internet to be fulfilled. Still other devices, such as firewalls and IDSs, interrupt traffic; we'll discuss these devices later in the chapter.

Attacking Your Network

Black hats use a variety of techniques to attack your network, depending on their goals. Network attacks often focus on gaining access to the network to see traffic or steal data. This means they must connect into the network between the packets your system sends and the destination of those packets so they can see the data being sent. Adversaries also often attack the network directly. These attacks usually revolve around trying to find ways to shut down network usage so the victim can't use their network the way they normally would.

Either way, the black hat's primary goal is to understand the network in the first place. Adversaries use many reconnaissance techniques to learn about their victims before they start their attacks. One common method they use is to do a *port scan*, which involves sending requests to every possible port on an IP address and then observing how the device at that address responds. Based on these responses, the attacker can determine a lot of information about the system. For example, if an IP address responds when the black hat scans ports 80 and 443, the attacker knows those ports are open and that the server is likely running web services of some sort. An adversary can use that information to attack the server directly or to trick other systems into thinking they're a friendly system. Port scans provide attackers with valuable information, making it much easier for them to craft different network attacks.

How Black Hats See Your Traffic

Packets provide attackers all sorts of details, including what devices they pass through, where a device is located, and what protocols a device is using, not to mention the data held within the packet. When an adversary (or anyone for that matter) intercepts traffic as it moves through the network, it's called *sniffing*. Like a hound dog on a trail, adversaries pick up bits and pieces of the traffic that travels through the network and reconstruct what they need from it.

On wired networks, sniffing can be difficult for attackers to accomplish, because networks are designed to send traffic to the intended recipient only. This means that the attacker must figure out a way to circumvent that design to make the traffic come to them. You'll see how they do this on wireless networks in Chapter 8. On wired networks, black hats can do this in a few different ways.

One method is by adding their hardware to the network. If you can connect your own physical device, the device can scan and copy that traffic as it passes through the network. So how does an attacker sneak a router or switch onto a network without anyone noticing? Although it would be difficult to do

that, adversaries often use a much smaller device called a *network tap*, which is designed specifically for this purpose. Figure 6-2 shows an example of a network tap. The tap connects to the infrastructure already in place on a network and copies traffic that passes through it.

Figure 6-2: A network tap (image altered from the original created by Andrew Fresh under the Attribution 2.0 Generic [CC BY 2.0] license, https://creativecommons.org/licenses/by/2.0/)

An adversary can also use a technique called *IP spoofing* in which they copy the IP address of a legitimate device on the network and imitate that device. Any traffic that was supposed to go to the device with the copied IP will go to the black hat as well. IP spoofing can trick you into connecting to a device, like a printer, that is actually an attacker.

A third method is to change where the traffic is being sent by altering the network settings. For example, by changing the default gateway on a device, a black hat can decide where traffic leaving the network goes. This allows them to direct traffic to a device they control so they can capture it.

Or an adversary could turn on *port mirroring* at a switch. Switches have numbered physical ports, or sockets you plug cables into. Typically, a switch has between 24 and 48 ports. Port mirroring tells the switch to copy all the traffic passing into or out of one port to another port. For example, if an adversary can turn on port mirroring at a switch, they can tell the switch to copy all the traffic coming in on port 1 to, say, port 22, where they've plugged in a device to capture it. Changing traffic settings usually requires a high level of access to accomplish, especially without network administrators noticing.

Another method an attacker can use is physically tapping the wire that traffic is passing through. How this is done depends on the type of wire used. For example, early networks often used a cable known as *coax*. It consisted of two copper lines wrapped in thick insulation. A special type of network tap called a *vampire tap* could pierce the insulation to physically connect two

metal prongs (the *teeth* of the device) with the two copper wires, allowing the tap to record any traffic that traveled across. Figure 6-3 shows an example of a vampire tap.

Figure 6-3: Example of a vampire tap (image modified from the original covered by the Attribution-ShareAlike 2.5 Generic [CC BY-SA 2.5] license, https://creativecommons.org/licenses/by-sa/2.5/)

Tapping fiber cables, which use pulses of light through glass tubes wrapped in insulation to send traffic, requires bending the cable and putting an unlit strain of fiber along the bend. When light goes through the bend, the unlit strand can grab some of the light, capturing the traffic.

The problem with these physical methods of traffic capture is that nearly all of them cause a loss in the signal along the cable. For example, bending the fiber cable increases the latency to the point that anyone monitoring the network would immediately realize something was wrong.

Man-in-the-Middle Attacks

Although using physical taps and changing network settings allow a black hat to see traffic in the network, making these techniques work requires a lot of setup. They're also difficult to hide, especially when they target larger businesses with dedicated IT staff who search for these types of attacks. Instead, adversaries can use a *man-in-the-middle* attack, which provides the same ability to read traffic without requiring physical access to the network.

In man-in-the-middle attacks, attackers place themselves in the traffic flow between their victim and the destination they're trying to reach. Instead of your traffic going directly to where you intended to send it—a web server, for example—it first goes to the attacker. The adversary can then read it, modify it, and pass it on to the destination. This allows attackers to capture your data and manipulate it for their own purposes. The worst part of these attacks is that they can be exceedingly difficult for the victim to detect. To the victim, everything looks like it's running correctly, albeit likely slower than normal. Figure 6-4 provides an example of a basic man-in-the-middle attack.

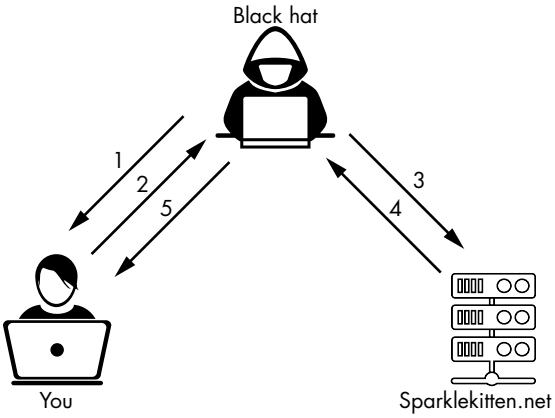


Figure 6-4: An example of a man-in-the-middle attack

In this scenario, the black hat sends you a phishing email with what appears to be a legitimate link from your bank ❶. When you click the link in the email, it takes you to the adversary's fake web server, where they've created a page that looks like your bank's website. You then enter your credentials on that website ❷. The adversary receives the traffic you send the website and modifies it so it appears to be coming from the attacker's computer rather than yours. The attacker then sends it to the legitimate bank site ❸ and gains access to your account ❹. The attacker then sends you a 404 Not Found error, so you won't realize what happened ❺.

Man-in-the-middle attacks can be executed using a variety of methods. In addition to the attack just described, adversaries could also create a proxy server and then trick the victim into connecting to the proxy. Recall that a proxy handles requests on behalf of another device, so by having the victim connect to the malicious one, the black hat can capture any traffic the victim sends to the internet. Another method of establishing man-in-the-middle sessions is changing where a victim gets their DNS information. If the adversary can trick the victim or load malware onto a system that changes the default DNS IP address, they can force the device to send all DNS queries to the malicious attacker's server rather than the legitimate DNS server. The malicious server can then respond with any IP address it wants, effectively allowing the adversary to decide where the victim's device sends web traffic.

Attackers can also use devices on the network to create a man-in-the-middle attack. If the black hat is able to access a device, they can change the settings to redirect traffic. IoT devices are especially susceptible to this type of attack. IoT devices are nontraditional devices connected to the internet, usually to provide some sort of enhanced feature. They include refrigerators, televisions, smart home assistants, and security cameras. These devices often have poor security, so attackers can easily take control of them. An attacker can sometimes update an IoT device's *firmware* (the code that runs the device's hardware) to include new code that allows them to capture traffic on the network. Because this method doesn't typically impact the device's function, it can be challenging to detect.

Denial of Service

Capturing traffic isn't an adversary's only option to attack a network. Black hats can also shut down the network entirely so no traffic flows out of it. This type of attack is called *denial of service (DoS)*. The basic premise behind a DoS attack is to stop the network from running its normal operations. For example, an attacker might send so much traffic to a single web server that no one else can access the server and the web pages it hosts.

There are many different ways to cause a DoS attack. As just mentioned, one way is to overwhelm a server with traffic so the system crashes. Ping packets, which were discussed in Chapter 2, are a great way to do this as well, because an adversary can change their sizes and send them in quick succession. In a *ping flood attack,* an adversary's device sends so many pings per second that the target device is unable to communicate on the network. Ping packets can fill up the system's memory to slow down the system. Ping floods are easy to execute, because they just require a system capable of sending pings and more bandwidth than the target system. Figure 6-5 shows a diagram of a ping flood attack.

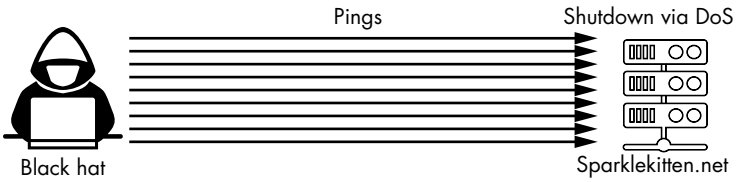


Figure 6-5: A ping flood attack

Another form of DoS attack exploits bugs in code to cause a DoS state. An example is known as the *ping of death*. Ping packets usually have a maximum size of 65,535 bits (a bit is the most basic unit of measurement for the size of data on a computer), but it's possible to create a ping packet larger than that limit. If a black hat can send a larger ping packet to a device, it can cause the system receiving the ping to lock up and shut down.

Ping flood and ping of death attacks are well known and much less common today than they used to be, as most of the vulnerabilities have long since been fixed. However, they're excellent examples of how to exploit conditions in code or network design to cause a DoS attack, and many modern DoS attacks use similar methods.

Distributed Denial of Service

A DoS attack occurs when one device attacks a single target, as in the ping of death attack. In a *distributed denial-of-service* (*DDoS*) attack, the attacker leverages multiple systems to attack a single target. By using more than one attacking system, the adversary can amplify the effect of the attack.

In the *Smurf attack*, an outdated example of a DDoS attack, the adversary began by spoofing their target's IP address. Figure 6-6 shows a diagram of a Smurf attack.

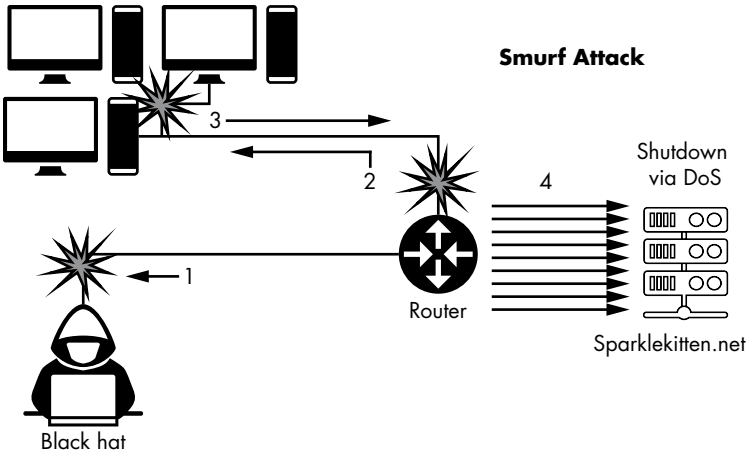


Figure 6-6: An illustration of a Smurf attack

After spoofing the IP address, the attacker sent a ping to a broadcast address on a large network ❶. *Broadcast addresses* automatically send traffic to every other device on the network ❷. The ping went out to all the devices on the network individually. They then responded to the target's IP address ❸. The target was overwhelmed with responses ❹, and it crashed.

A more modern example of a DDoS attack is the *DNS amplification attack* (Figure 6-7).

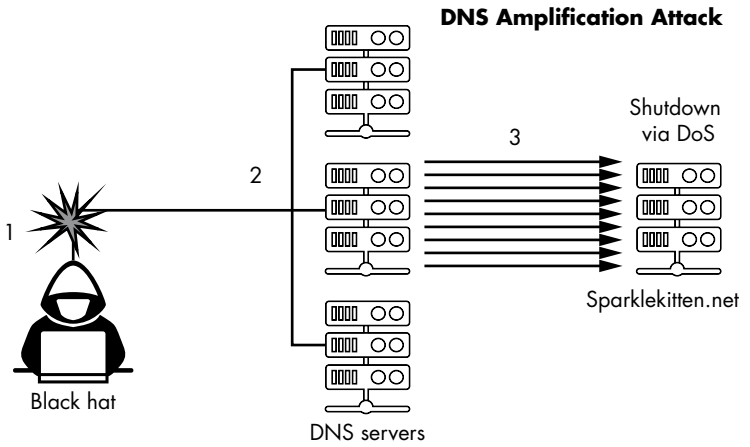


Figure 6-7: A diagram of a DNS amplification attack

Similar to the Smurf attack, the DNS amplification attack uses basic DNS requests to overwhelm a victim's connection to the internet. The black hat crafts DNS query requests that spoof the victim's IP address ❶. These queries also include large response parameters, meaning they'll accept the largest possible size of a response to a single query. The adversary then sends a constant stream of these queries to publicly available DNS servers ❷. Although the queries are relatively small, the responses are large. The DNS servers

send these large responses to the victim's IP address ❸, causing a DoS state. Attackers used this type of attack in 2016 against the security website Krebs on Security, causing one of the largest DDoS attacks seen up to that point.

One way adversaries attempt DDoS attacks is by creating a *botnet*. A bot, in this context, is a system that the black hat has compromised so it accepts commands from an attacker-controlled server. Compromising a device in this way usually requires installing malware or malicious firmware. Attackers can turn hundreds of thousands, sometimes millions, of devices into bots that can all receive commands at the same time. This botnet can send simultaneous requests to a server to produce a powerful DDoS attack. One of the larger botnets recorded, the Mirai botnet, is believed to have infected 600,000 IoT devices at its peak. Each of those devices could be used to send out pings, DNS queries, or other types of DoS attacks without the attacker having to directly attack their target. This makes botnets highly effective tools for shutting down a victim's system.

Defense Against Network Attacks

Defending against network attacks requires a keen understanding of how your network is laid out and what resources are attached to it. It's much easier for a black hat to exploit an unorganized network, because IT administrators will have a harder time ensuring the right settings and security controls are in place if they're not sure how traffic is flowing through the network. This is especially true for large networks that might have thousands, or even tens of thousands, of systems.

One way to solve this problem is to organize your network into zones and base your security around each zone rather than each system. When a system is added to a zone, you'll know that it needs to have a certain set of security controls in place to follow the zone's requirements. A common type of network zone used when systems are accessible from the outside is called the *demilitarized zone (DMZ)*. The DMZ sits between the internal private network and the external public network. It's sort of a hybrid of the two. Typically, the administrator places systems that allow people to connect from the outside in the DMZ. For example, if you're hosting a website on a server, you'd put the server in the DMZ. The DMZ usually has strict controls to ensure that traffic is monitored as it flows in and out, so no attacks or exploits can breach it. Figure 6-8 shows a diagram of a DMZ.

You can break up a network into as many zones as necessary to maintain security controls. For example, you might have an external network zone, a DMZ, and an internal zone that includes subzones for your database and HR systems. The only drawback to using numerous zones is managing and updating all the levels of controls for each one. Sometimes, your systems might also seem to fit into multiple zones, so you'll have to consider carefully where to place them.

Once you've established your network zones, you can add controls.

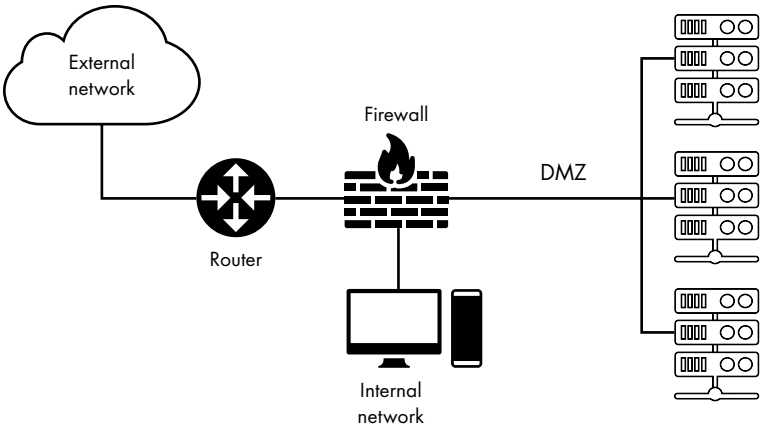


Figure 6-8: Example placement of a DMZ on a network

Firewalls

One of the most fundamental controls for a network is a firewall. At a basic level, a firewall regulates traffic in one of two ways: it either *allows* traffic to pass through and continue on to its destination or *denies* traffic and blocks it from moving on. This decision is made by matching the traffic to a set of rules to determine how it should be treated.

Many types of firewalls are available. They can be either software-based or hardware-based. *Software firewalls* are computer programs; typically they're used on devices running other software, such as a server running a website or a desktop computer. Software firewalls are one of the last lines of a system's defense. Usually, they have fewer features than a hardware firewall but are still necessary to help regulate which applications on a system are allowed to send traffic on the network. *Hardware firewalls* are physical devices used to view and regulate traffic that flows into the network. In general, hardware firewalls are more robust than software firewalls. Depending on the model, they offer many security features and the ability to make additional choices other than just allowing or denying the traffic.

Packet-Filtering Firewalls

Software firewalls use a *packet-filtering* method. They look at packets as they come into a system and determine whether or not to allow that traffic, based on a set of rules. As discussed earlier, packets contain information about the data that is being sent, including the destination and source IP addresses, and the port number they're arriving on. Packet-filtering rules are usually based on the source IP address and the port number being used. Table 6-1 shows examples of firewall rules.

Table 6-1: Example Firewall Rules

Rule number	Allow/deny	Protocol	Destination IP	Source IP	Port number
1	Allow	Any	192.168.15.1	Any	80
2	Allow	Any	192.168.15.1	192.168.15.2	23
3	Deny	Any	192.168.15.1	Any	23
4	Deny	Any	Any	Any	Any

Each row represents a rule in the firewall. In the Protocol column, *Any* means it doesn't matter what type of network protocol is used. Sometimes you'll see the wildcard symbol (*) used in place of Any, but it means the same thing.

Notice that the second and third rules use the same port number. However, one rule allows a specific source IP, whereas the other denies any source IP. It's important to note that when a firewall reads its rules, it goes through them one by one until it finds a matching rule. If the rules aren't in the proper order, it can cause problems with how traffic is managed. For example, if you reversed rules 3 and 2, the rule denying traffic on port 23 from any source IP would be placed before the rule allowing traffic from a specific source IP; so when traffic came in from the IP address 192.168.15.2, the firewall would deny it, because that IP address matches the source IP of Any. For this reason, it's essential to place rules that allow traffic, especially those that list specific IP addresses, before rules that deny it.

The fourth rule ensures that if no other rule applies to the traffic, it's denied. We call this an *explicit deny*. Many modern firewalls include a "deny all" rule by default without you having to add it to the list of rules; this is known as an *implicit deny*.

Stateful Inspection Firewalls

Another popular form of firewall is the *stateful inspection firewall*, which is typically only used to regulate traffic coming into a network. Like a packet-filtering firewall, a stateful inspection firewall has rules that determine how to manage traffic based on factors like IP addresses and port numbers. In addition, the stateful inspection firewall considers the conditions to determine how to apply the rules.

In other words, when traffic comes through the stateful inspection firewall, the outside device makes a connection to the firewall. This differs from a packet-filtering firewall, which views the traffic as it passes through other devices. The firewall learns how the connection was made and certain parameters about it—for example, if it uses encryption. It also determines whether the traffic should be allowed or denied. If the traffic is allowed, the firewall monitors the traffic as it moves to see whether the state of the connection changes. This allows the firewall to track packets by certain details, such as source and destination IP address. When packets from the same location come in at a later time, the firewall can use those details to determine whether that traffic is allowed without having to completely reexamine it.

Application Firewalls

Application firewalls offer protection designed for specific types of applications, such as web servers or databases. These firewalls include special security controls that help defend against attacks targeting the applications they are protecting. The application firewall can also inspect traffic at a much deeper level, allowing it to see more information than even a stateful inspection firewall.

For example, if you're using a *web application firewall (WAF)* to protect a web server, it will inspect all the HTTP requests sent to that web server. A WAF can see attempts to send malware, vulnerabilities being exploited, or even deviations from the typical web server setup. Application firewalls also have more options regarding how to handle traffic. For instance, some can send the traffic to a new IP address to be analyzed. Many application firewalls are bundled together with other security appliances, such as intrusion detection systems (discussed in the next section).

The main drawback to an application firewall is that it can be very slow and resource-intensive. Scanning a packet deeply and analyzing the data for violations takes a lot more time than simply checking the IP address and port number, the way a packet-filtering firewall does. It also requires a lot more memory and processing power, which means systems that run application firewalls are more expensive than systems that run packet-filtering firewalls. Higher levels of security controls require this trade-off. It's also important to know that application firewalls are specific to their designated application; you can't put a WAF in front of a database server and expect the same level of security performance.

Firewalls can't read encrypted traffic, because the traffic is, well, encrypted. However, when traffic is encrypted on the network, the packet header, which contains the IP address and the port number, is often unencrypted. This means stateful inspection firewalls and packet-filtering firewalls work normally, whereas application firewalls would need to decrypt the data in the packet to examine it before they can decide to allow it or not.

Intrusion Detection Systems

Although firewalls are necessary to prevent many unwanted connections, they're not always efficient at detecting attacks or finding malicious traffic hiding inside legitimate traffic. Because the firewall follows the rules as written, it might sometimes allow traffic from a black hat if that traffic matches an allow rule. To catch these attacks, you should add an IDS to your network.

IDSs are designed to detect attacks happening on the network or system, and they provide alerts to security personnel. An IDS can detect attacks that might be difficult for other security controls to handle. For example, they can detect a ping of death or Smurf attack, whereas a firewall might accept that traffic if it has a rule that allows ping traffic. Like firewalls, an IDS can be software- or hardware-based and can monitor a single system or a portion of a network. An IDS is often used in critical areas of the network, such as at the entrance to the DMZ or the internal private network.

To detect attacks, an IDS uses two different methods: signatures and heuristics. *Signatures* are similar to firewall rules but might include additional behavioral elements, such as the time that traffic comes in or what type of connection it's trying to make. The main characteristic of signatures is that they're rigid. Once you write the signature, the IDS will only look for traffic that exactly matches that signature. For example, if you write a signature that looks for specific requests to sparklekitten.net and a request comes in for sparklekitten.us, the signature won't detect that request. Signatures can also detect known malware.

A heuristics-based IDS monitors the system or network it's on to learn what a normal baseline level of traffic looks like. A security professional then sets the heuristic system to alert them whenever certain conditions change on the network. For example, if the number of connections to a web server normally hovers around 1,000 a minute, the security person might set a threshold of 10,000 a minute to set off an alert because this increase in traffic could indicate that either a hot sale or a potential DoS attack is occurring. Heuristic systems are very efficient at detecting new attacks, because they don't have to rely on a signature to determine that an event is malicious. Experts must analyze malware to create malware signatures, so if the malware is brand new, it's likely that a signature system won't detect it. But heuristic systems also require constant fine-tuning to ensure they have a correct baseline of normal traffic. An alert shouldn't go off whenever the system makes 10,000 connections a minute if your server does so on an average day.

Intrusion Prevention Systems

Typically, an IDS is a passive system. It can send alerts but doesn't stop the malicious traffic. Similar to application firewalls, it can take IDSs a long time to examine data and make sure it doesn't match a signature or heuristic characteristic. To improve their speed, many IDSs copy the traffic and analyze it after it has passed through the system. But this means the traffic has already reached its destination by the time it's detected as malicious.

Obviously, this can be a major security issue, especially for malware that might infect the system before a security person can react to stop it. To help prevent this type of problem, security researchers created devices called *intrusion prevention systems (IPSs)*. An IPS works similarly to an IDS in that it uses signatures or heuristics to detect malicious traffic. But instead of being passive, the device actively interacts with the traffic before it moves on to its destination to prevent it from harming its target.

Depending on the model, an IPS has several ways of handling malicious traffic. One method is to just block the traffic entirely. The IPS might do this directly, or it might change a firewall rule to block the traffic. The IPS can also send the traffic to a special security system where the security team can analyze it to learn an attacker's technique. An IPS can also remove or quarantine malware before passing on the clean traffic. In addition to these proactive measures, IPSs send alerts, just like IDSs.

Much like application firewalls, the main drawback to an IPS is that it is much slower and requires more resources than an IDS. This makes an IPS more expensive than an IDS. Therefore, you typically see IPSs used only in critical locations, like the entrance to the DMZ, whereas an IDS might be on every server inside the DMZ, as shown in Figure 6-9.

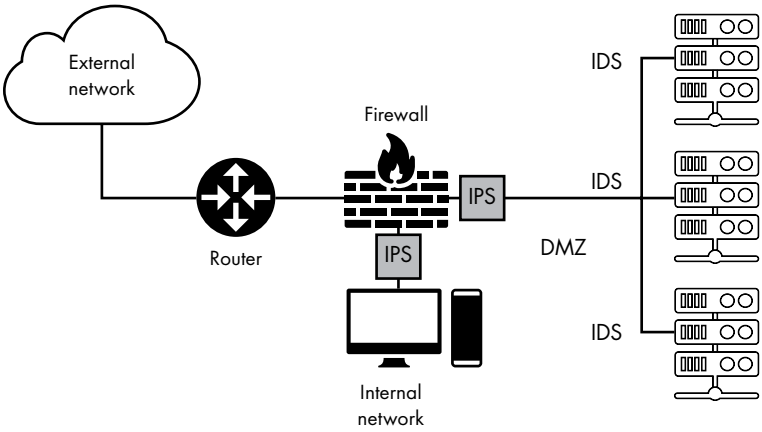


Figure 6-9: Example placement of IDSs and IPSs on a network

IPSs and IDSs are often bundled into all-in-one appliances, which combine several different services into one system. These appliances can include a slew of security devices, including firewalls, email filters, proxy servers, and more.

Exercise: Setting Up Your Firewall

Both Windows and macOS include built-in firewalls that you can use to block traffic to specific applications coming into your system. Although both have a set of default rules that provide a good amount of security, you can also add, remove, or modify these rules. Customizing the rules can help you keep your device even more secure, especially when adding new applications to which you don't want to allow external connections. In this exercise, you'll configure your firewall, adding an inbound rule to secure a new application you've just installed.

Windows

To access the Windows firewall, enter **firewall** in the search bar at the bottom-left corner of the screen. The Windows Defender Firewall with Advanced Security app will appear. Click this app to open a window that displays general firewall information and settings. On the left side, click **Advanced Settings** to open the window shown in Figure 6-10 (you must have an administrator account to do this).

Figure 6-10: Windows Defender Firewall advanced settings

The left-hand menu shows the kinds of rules you can set. You should see submenus for inbound and outbound rules, as well as a few other options. When you click one of these menu items, the options for it will display in the middle box and the actions you can perform will appear on the far right.

For now, you should see the firewall profiles in the main box. In Windows Defender Firewall, profiles can change the firewall's behavior depending on what type of network the system is connected to. This is very helpful for devices like laptops that might switch between different networks often. The *domain* profile is for devices that are remotely managed, such as in a company. The *private* profile is for private networks, like your home, and the *public* profile is for public networks, like a wireless network in a café. The options for each profile are the same by default, but you can change them. Just be careful when making any modifications, because the changes affect the entire system, not just specific applications.

Now that you've looked over the profiles, let's add a new rule. Choose an application to which you want to apply a new rule. Having an application in mind helps you set the rule's parameters. The main factors to consider are what port numbers the application uses and what type of traffic should be allowed to connect to that application. For example, if you're installing a new game, you'll need to open the port numbers that game uses. You'll also need to check whether there are any particular protocols or types of traffic that must connect to the game for it to work. For most applications, you can find this information in the user manual or the website's help section.

With an application in mind, you need to select the menu in which to add the rule. For this exercise, let's add it to Inbound Rules. Inbound rules apply to traffic that is coming into your system, whereas outbound traffic is data you send out from your computer. Click **Inbound Rules** on the left to bring up the current list of rules for inbound traffic. Figure 6-11 provides a selection of those rules as an example.

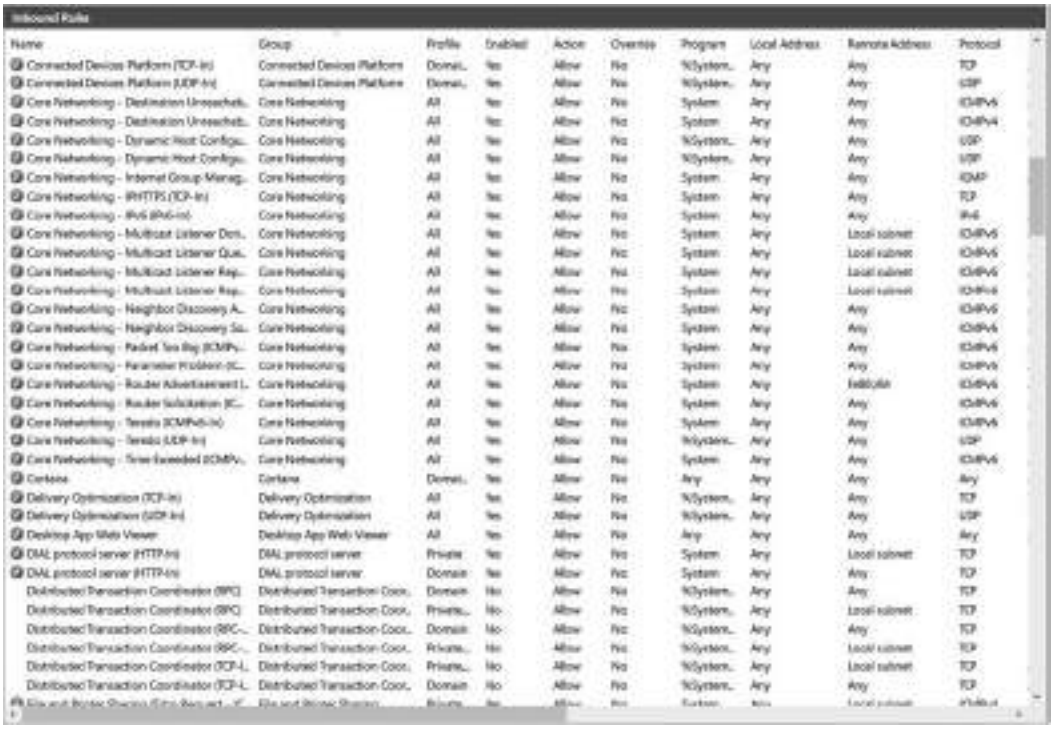


Figure 6-11: List of inbound rules

This window is laid out a bit differently from Table 6-1 in the section on packet-filtering firewalls earlier in the chapter, but you can see that the same elements are present. A green check mark icon indicates that the rule is allowing traffic; rules without icons are currently disabled. To the right is the name of the rule and the *group* it belongs to, which categorizes the rule based on which protocols or applications it relies on. For example, in Figure 6-11, you can see several rules that deal with some of the Desktop's core networking functions, and thus are aptly placed in the Core Networking group. The next column is the profile that the rule applies to. As you can see, some rules are active only in certain profiles. Subsequent columns deal with what the rule does, including the action (allow/deny), the application it applies to, the IP addresses it applies to, and the type of protocol it applies to.

The protocols usually relate to the type of connection used, such as TCP. If your application doesn't use a specific protocol, it's safe to use Any here. The next column, not shown in the figure, relates to the port numbers the rule applies to. Recall that port numbers can tell a system about the type of traffic being sent. Most applications use common port numbers, such as port 80 for web traffic. Others might use ports that are unique to that application; for example, the game *Doom* uses the port number 666 for its multiplayer mode. When installing a new application, you might need to add a rule to allow it to use a new port number. Let's do that now!

On the right side of the Inbound Rules window, click **New Rule**. This should bring up a wizard to help you set up the rule (Figure 6-12). The first

options you see are for the Rule Type. Program is used to make rules for a specific application. You could also use Port, which is for rules that affect specific port numbers; Predefined is used for connections to one of the groups created by default, such as Core Networking, and Custom creates a rule with your own parameters in place instead of limiting you to either port number or application. Select **Custom** to see the available options.

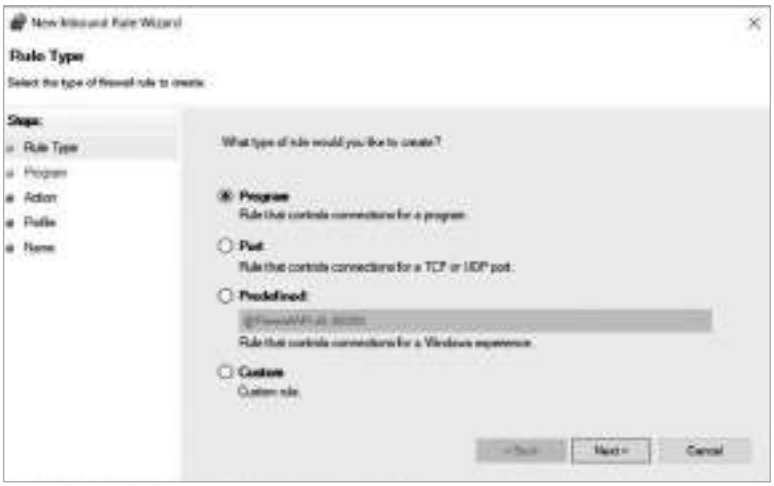


Figure 6-12: Rule Type options

Specify whether you're creating a rule for all programs or for only a specific application. To do that, select **This Program Path:**, click **Browse**, then find where you saved the program on your system and select it from the menu. In Figure 6-13, I used a fake program name as an example. Feel free to do the same, or keep the All Programs option selected. Then click **Next**.

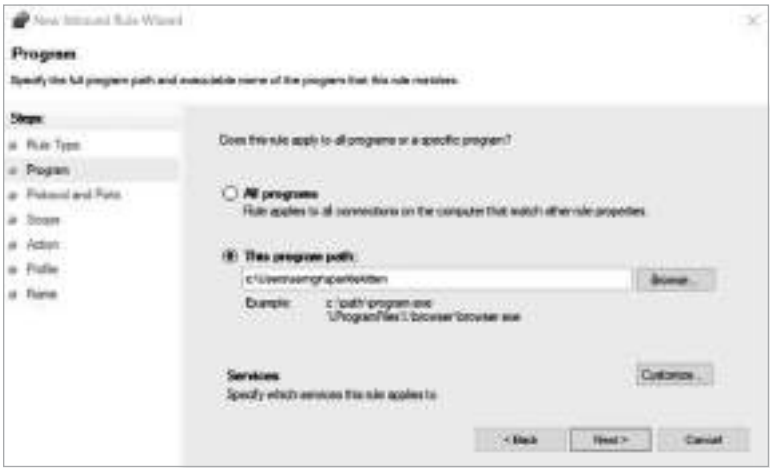


Figure 6-13: Example of program selection

Now list which ports and protocols you need for your rule. If you're making a rule for an application, you should be able to find out which ports and protocols it uses from its company's website or support materials. In this example, I used port 80 and the protocol TCP to represent a web service. Figure 6-14 shows the options filled in.

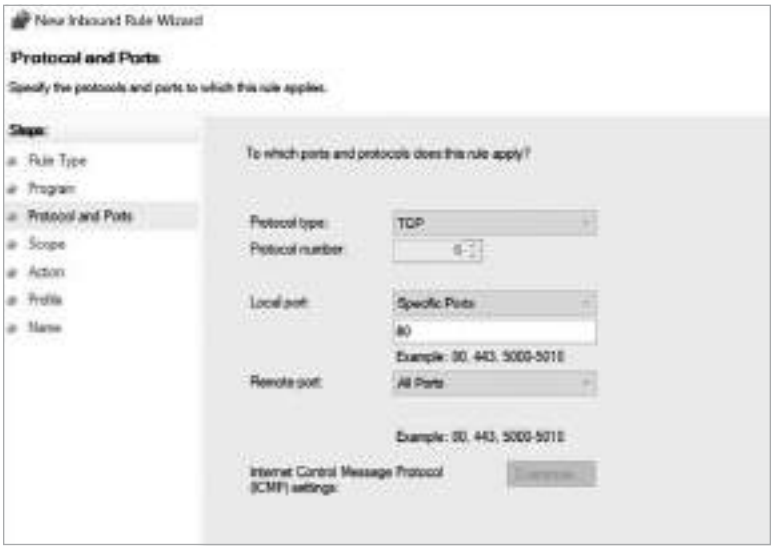


Figure 6-14: Example of port number selection

Click **Next** to specify the IP addresses the rule applies to in the Scope step. Unless you're trying to communicate with a specific device, for example a printer, it's generally best to leave a broad scope of addresses accessible so you don't have to constantly add new ones. This is also a good time to rethink the rule and make sure you're comfortable with allowing all sources of traffic in or out.

Click **Next** to go to the Action step and decide on the action you'll take. You have three options: allow, allow if secure, and block. The allow if secure option lets a connection through only if it's using a well-known security protocol that includes encryption (we'll discuss some of those protocols in Chapter 9). In this exercise, I made the rule block the connection, as shown in Figure 6-15. This means any traffic on port 80 trying to access the specified application will be blocked, effectively isolating it from web traffic.

After selecting the action, click **Next** to choose your rule's profile. Generally, you should select all three profiles so the rule works no matter what type of network you're on. That said, you might choose a specific profile type to give your computer a different level of security. For example, you might block a connection to a public network that you'd normally allow on a private network.

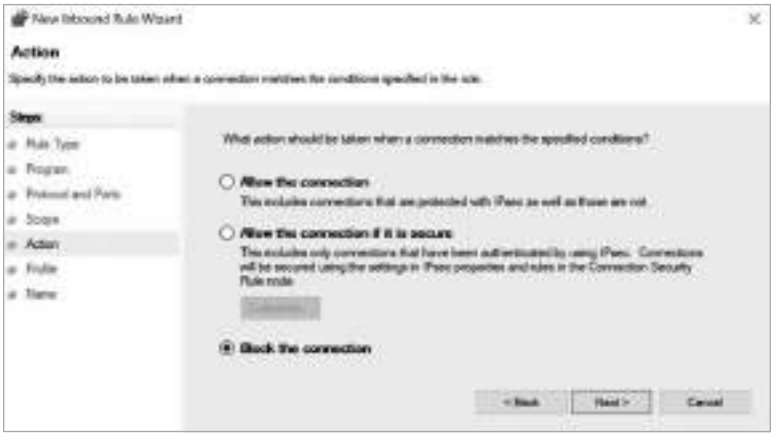


Figure 6-15: Setting the connection type

Finally, click **Next** to name the rule. Then click **Finish**. You should see your new rule at the top of the list in the Inbound Rules window, as shown in Figure 6-16.

Figure 6-16: The final rule in place

Now that you've added a new rule, you can easily create more rules to fit your network's needs while also keeping your device more secure. Remember that when you no longer need a rule, you can disable it rather than deleting it in case you need it in the future.

macOS

macOS includes a built-in firewall that you can activate. Although its options aren't as robust as Windows Defender Firewall's, it still provides great protection once it's set up correctly. To find the firewall, click the Apple symbol in the top-left corner of your screen, and then click **System Preferences**. Click **Security & Privacy**, and then click **Firewall** in the top menu bar. Figure 6-17 shows the resulting dialog. If the firewall is turned off, you'll need administrator privileges to turn it on. Click the lock at the bottom of the dialog to change the firewall settings. Click **Turn On Firewall**, and then click **Firewall Options**.

In the options dialog (Figure 6-18), you have a few choices. You can block all incoming connections to the system. Although this will increase your security, your applications won't work properly if they can't accept incoming connections. You can also add applications and specify whether they should accept or deny incoming connections. In Figure 6-18, Adobe Photoshop is set to allow all connections.

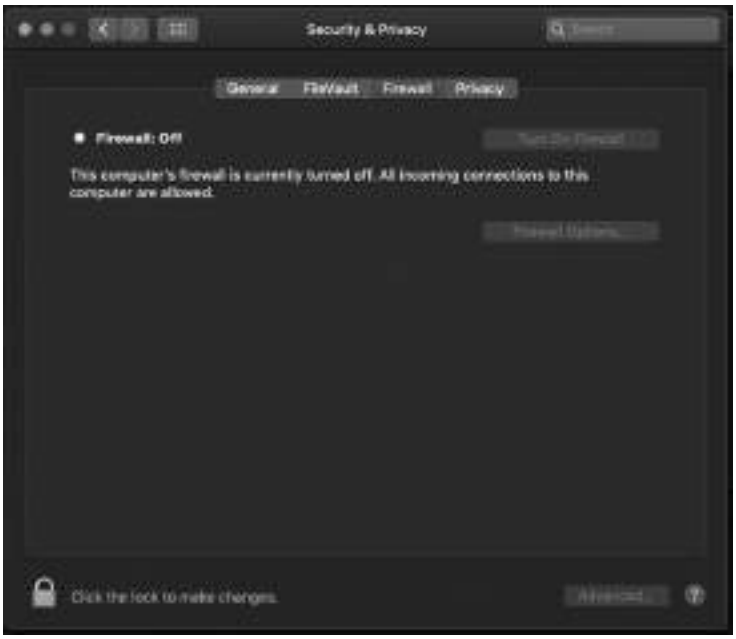


Figure 6-17: macOS firewall settings

Figure 6-18: macOS Firewall options

The next option allows built-in software to accept incoming connections automatically. This means that any Apple product installed on the system will accept incoming connections by default. Apple has many processes

in place that test its software for vulnerabilities, but you should be aware that there remains a chance that a black hat could exploit this firewall condition to gain access to an application.

The next option is similar but applies to downloaded signed software. *Signed software* is from a verified legitimate source. The last option, *stealth mode*, prevents your device from sending an answer when it receives certain kinds of traffic, such as a ping packet. This can be helpful for blocking an attacker from learning information about your device, including whether or not it's actually on the network.

Unfortunately, macOS doesn't offer as many options as Windows Defender Firewall in the GUI; you can add applications to your firewall list, but you can't create custom rules with port numbers using the options in System Preferences. There's also a firewall called *pf* that runs as part of the operating system; however, to access it, you'll need to access the configuration file *pf.conf* in the *etc* folder. You can find a manual for it by entering the `man` command, **`man pf.conf`**, in a Terminal window, which brings up help documents built into the system.

I highly suggest you read the manual thoroughly before making any changes to the firewall. Unfortunately, the specifics of configuring *pf* are outside the scope of this book.

Now that you understand how to make rules for Windows and macOS, you can fine-tune your firewall to secure your system while you work. The important detail to remember is that creating a well-tuned firewall takes time and some trial and error. It's also not a feature you can just set once and not think about again. Reviewing firewall rules to make sure all your applications are protected is essential to ensure you maintain your system's security.

Conclusion

After reading this chapter, you now know that prevention is critical to protect your network's security. Your security efforts should primarily focus on preventing an attack from occurring rather than trying to stop it once it begins. By setting up a firewall with the correct rules, you can thwart many low- and mid-level attacks just by stopping the traffic from entering your network. For high-level attacks, using an application firewall or an IDS or IPS can help detect an attack before it has a chance to reach its destination or unleash its payload. These systems can provide security experts with the crucial time they need to react before a black hat gains access.

7

ATTACKS IN THE CLOUD

You've heard the term *cloud computing*, either listed as a tech product feature or in commercials. You might have used cloud storage to save photos or documents to the internet. So you can understand why black hats who want access to these files would attack your cloud storage.

But these brief descriptions don't explain what cloud computing means. In this chapter, we'll discuss how cloud computing works and the basic setup of cloud computer services. We'll examine how attackers target the cloud and steal the information or services hosted there. Then we'll explore what you can do to better secure your cloud accounts, as well as be more informed when choosing a service to use. In this chapter's exercise, you'll use some of the same techniques adversaries use to perform an attack called *SQL injection*; that way, you'll gain a better understanding of how they work. By the end of this chapter, you'll know how the cloud operates and what you can do to protect your cloud storage from attackers.

How Cloud Computing Works

The term *cloud computing* can be confusing because it's used in many different contexts, especially when it's used to market products. At its most basic, cloud computing just means *using someone else's computer to do something*. At first, that definition might seem impossibly vague. But that's sort of the point; cloud computing encompasses a range of services and systems, some of which you might not even know about.

One example of a cloud service you're probably familiar with is website hosting. As discussed previously, when you go to a website, you're reaching out to a server that hosts that web page. Theoretically, this means that anyone who wants to have a website has to have their own web server, which they set up and maintain. But in reality, tons of companies provide web hosting services so you can have a website on the internet without having to own a web server. Some of these services, such as Wix, Google, and Squarespace, offer additional cloud services, such as database storage, marketing ads to other websites, and even website-building tools.

One main advantage of a cloud service is that you don't have to maintain the equipment and systems required to use a certain piece of hardware or software. If I wanted to host a website that sells the t-shirt designs I doodled on the back of my napkin at lunch, it would be difficult for me to raise the money necessary for all the equipment required for a modern website to operate effectively, never mind the know-how to set it up and run it. Instead, the cloud service allows me to use its equipment and expertise. This reduces the overall cost of the service, which is split between the service's users.

A downside of using a cloud service is that you have limited control over how you can operate the equipment. Using the website service as an example, the service offers you no control over what type of web server the company is using. You also have limited control over the features the service provides. If it doesn't offer a particular service—for example, an online store for your website—it can be difficult to get the service to add that feature. You're limited by your contract for the services too. For example, you might have a contract that allows only 100 visitors to your website a day. If more than 100 visitors browse to your site, either you'll pay a premium price or the additional visitors might be unable to access the page.

As the internet has expanded and the number of network-connected devices has grown, so too have the offers of cloud services. Understanding how each service works can be confusing. One way of categorizing cloud systems is to describe what type of service they offer. Typically, we label these as the service name followed by *as a service*. For example, you might have *AaaS*, which could mean *Application as a Service* or *Analytics as a Service*. As you can see, cloud computing encompasses so many features and functions, even the acronyms have multiple definitions. In fact, people sometimes use the acronym *XaaS* to mean *Anything as a Service*, which shows how widespread cloud services can be.

For simplicity, let's look at the three main services we usually associate with the cloud.

Software as a Service

Software as a Service (SaaS) is software hosted on someone else's system that users can access through a network connection, typically by logging in through a web portal.

A great example of SaaS is Microsoft Office 365. This product allows you to access Microsoft Office applications, like Word or PowerPoint, through a website rather than installing the applications on your computer.

SaaS has a few advantages: it gives you access to the latest software updates as well as the ability to use the software from many different systems. Also, SaaS often provides integration with other cloud offerings. For example, Office 365 integrates with OneDrive, which is Microsoft's cloud storage service.

Platform as a Service

Platform as a Service (PaaS) includes all the infrastructure, tools, hardware, and software needed to run a service. A web hosting platform is one example of PaaS. Another example is email services, like Gmail or Outlook. The platform in this case is an email system, which requires a server, website portal access, applications and protocols, and more to operate. Just like with SaaS, but on a larger scale, you get access to the entire platform without having to manage the systems required to run it.

PaaS doesn't give you as much freedom as SaaS, because you're essentially limited by what the cloud provider runs. You have limited choice in the type of hardware or software to use as part of the platform. The trade-off is easier-to-manage options along with more robust features than you might have if you ran your own applications or hardware.

Infrastructure as a Service

Running software or even a platform is fine for small projects, but what if you need a large amount of resources to run multiple different types of platforms at once? This is where *Infrastructure as a Service (IaaS)* comes into play. IaaS provides all the resources needed to maintain the services you or your company provide to customers. Again, that description may seem vague, but this is because IaaS typically covers large-scale operations that include multiple components working together. Let's look at an example of a conventional IaaS setup.

Let's say you create a video game that includes a multiplayer feature. The feature lets players join the same game to build forts to defend against attacking monsters. You're a small indie developer, so you mainly focus on creating the game and its art design. However, you know you'll need a lot of systems to run the multiplayer component, so you hire an IaaS company to provide the necessary systems.

The IaaS company will provide not only the servers for you to host your game, but also the equipment necessary for players to connect to the servers when they want to play the game. In addition, the IaaS will provide storage

to hold all the players' data when they're not playing so they can retain their game items. The IaaS might also provide help desk staff in case players have trouble accessing a game or a server crashes.

An IaaS provides lots of equipment and expertise designed to manage an essential service within a company. Another popular form of IaaS, known as a *managed service provider (MSP)*, provides most of the technical equipment for a company, such as desktops, servers, and routers, as well as management in the form of technicians and help desk staff.

IaaS gives you the least flexibility when it comes to what sorts of systems are available and how much control you have over them. In the multiplayer game example, you, as the game's owner, wouldn't necessarily be able to tweak operations related to how the systems are run, such as how often servers are patched, what type of security is in place, or how systems are recovered should they break or crash. Be sure to pay particular attention to the contract you sign with an IaaS, because the IaaS will be a major component of how you work.

Figure 7-1 shows some examples of services provided at the various levels of cloud offerings.

Software as a Service (SaaS)	Office 365 Salesforce Electronic medial record applications
Platform as a Service (PaaS)	Squarespace Gmail OneDrive
Infrastructure as a Service (IaaS)	Amazon Web Services Microsoft Azure Google Cloud Services

Figure 7-1: Application examples by cloud levels

Security as a Service

Another field in cloud computing is *Security as a Service (SECaaS)*, which provides cybersecurity services to protect clients from attacks. Security as a Service can include a number of different components. Some are limited in scope, such as vulnerability scanning services, and others are full-security services with personnel on call in the event of an attack. One example, known as *managed detection and response (MDR)*, attempts to detect attacks happening in real time and respond to them in a way that prevents further damage. The service can also include ongoing security work, such as penetration testing or forensics.

Attacking the Cloud

Many of the attacks discussed in previous chapters are effective against cloud infrastructure. For example, social engineering, discussed in Chapter 3, and

authentication attacks, discussed in Chapter 5, are very strong attacks against cloud services. Because many cloud services offer websites through which a client can interact with the service (for example, a Gmail account or Office 365 application), social engineering can help a black hat gain access to that account information and log in as the user. It can also be more difficult to detect these malicious logins, because the cloud application is so widely accessible that checking for typical indicators of malicious activity, like the time of day or location of the login attempt, might not work.

Another example of an attack that works against cloud computing is the man-in-the-middle attack, introduced in Chapter 6. If an adversary can intercept network traffic between the client and their cloud service provider, they can manipulate the traffic to access the service or steal the data that is being transferred. Setting up a fake page that looks like the cloud service login page is one way that an attacker can create this man-in-the-middle scenario.

Malware is also a viable option for attack. Many adversaries have targeted cloud services with malware because of the amount of resources they can exploit. *Crypto-mining malware*, which illegally mines cryptocurrency, like Bitcoin, is one example. Crypto-mining takes a lot of CPU resources and power, so adversaries sometimes attempt to use cloud service resources to mine cryptocurrency. This kind of attack can be tricky to detect, because many crypto-mining programs are disguised as legitimate applications running in the cloud environment.

Most cloud attacks aim to gain access to the infrastructure the cloud provider runs. One way to do this is to look for exploits in the client-facing systems the cloud service provides, and then use those exploits to access the private, internal resources. We often refer to these exploits as *web application attacks*; we'll take a closer look at them next.

Web Application Attacks

Many cloud services have a client interface that you can access through a website or other online application. This interface allows you to interact with the features you paid for. It also provides a perfect place for adversaries to attack. Because the application must remain available for clients to use, it can be difficult to stop an attacker from accessing it, too. Some web applications are open to the public as well, meaning anyone, at any time, can access them.

Also, web applications are often connected to resources inside the internal network. This enables outside users to access data that isn't usually available to the outside, such as user profiles or personal information. It also creates a way for adversaries to gain access to the inside network and steal that data.

As discussed in Chapter 6, private networks usually send connections from the outside network to the DMZ. This allows users to access cloud services and other resources without having access to the internal network. When a user needs access to the data stored on the internal network, the cloud service in the DMZ accesses those resources on the user's behalf.

Yet there are many ways that an adversary can attack the internal network through external cloud services. One of the main goals of these attacks is to gain *arbitrary code execution*, which is the ability to execute whatever commands or code you want on a system. Usually, only accounts with the highest privileges can do this, but there are often ways to bypass the restrictions in place. Once a black hat can run their own code, they can fully infiltrate the system by either installing a backdoor or creating an account.

Let's look at some of the most common examples of web application attacks to determine the general structure of this type of attack and how they work.

Buffer Overflows

A *buffer overflow attack* attempts to fill a computer's memory so that either the system crashes or the attacker gains access to normally unauthorized parts of the system. To understand how this attack works, you first need to know how memory allocation works. A system uses memory to store all of its information. Normally, a system only has a set amount of memory, which it allocates in blocks to its functions. Applications, such as those cloud services use, are given a set number of blocks of memory to use. Unused memory is known as a *buffer*.

When attackers commit a buffer overflow attack, they intentionally try to overflow the memory buffer allocated to the application. For example, if an application were given five blocks of memory, the adversary could attempt to inject six blocks of data into it to overflow the allocated space. Figure 7-2 illustrates this process; the word *excessive* is larger than the space allocated in section A.

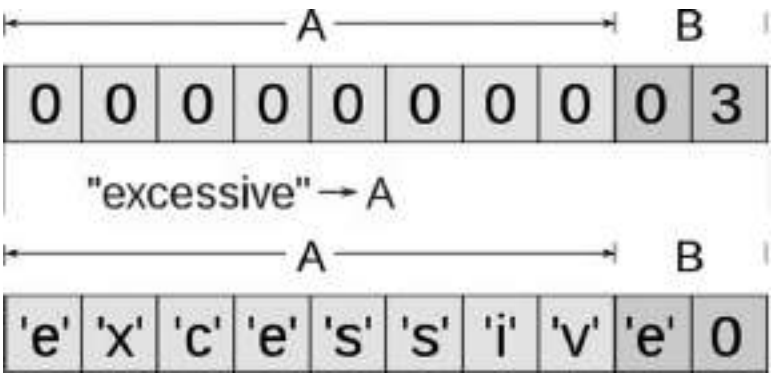


Figure 7-2: Buffer overflow example (image modified from the original covered under the Attribution-ShareAlike 2.0 Generic [CC BY-SA 2.0] license, https://creativecommons.org/licenses/by-sa/2.0/)

Once the buffer is breached, a few different events can happen. Some systems will cause an error and crash because they can't store all the data provided to them. Another option is that the data overflows into the next block.

Often, attackers place commands at the end of a long string of data that they submit to the application. If the memory overflows the buffer

and runs into a memory block allocated to administrative functions, the system command might be executed using a high level of privilege. This could result in the adversary executing any code they want to on the system.

The best way to manage a buffer overflow is to use a feature called *input validation*. When a user sends data to the system to be committed to memory, input validation code first checks to ensure the data will fit and arrives in the proper format. If the input is unacceptable, the system drops it before it commits it fully to memory. This stops the buffer overflow attack before it can begin. Another technique that application developers can use is called *fuzzing*. Fuzzing generates random inputs of various lengths and automatically applies them to the input fields on a website or cloud application. This allows a cloud service to ensure that input validation is in place and that no exploits, like a buffer overflow, will occur.

SQL Injection

Structured Query Language (SQL) is a programming language used to access or manipulate the data stored in databases. Web applications sometimes use SQL databases to store information that the website's users can access. For example, a website that has a login feature might use a SQL database to store its users' account information, including their passwords. When a user logs in to the website, the web application creates a custom SQL query and sends it to the database to retrieve the user's information. Figure 7-3 shows how this works.

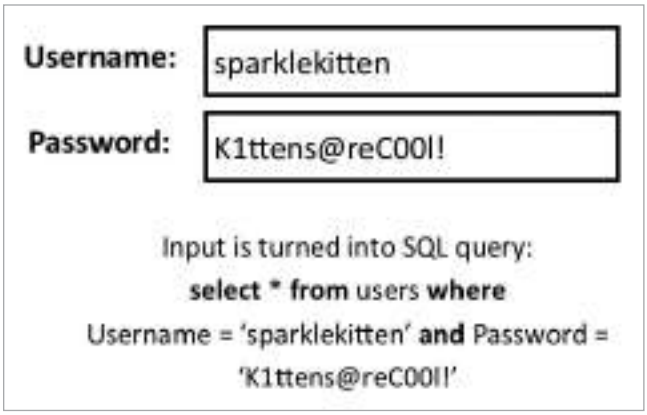


Figure 7-3: An example of a normal SQL login

Sometimes, using a SQL query to access a database means that a user can use the password input field to send any values of their choosing directly to the database for processing. For this reason, it's possible for an attacker to inject a text string other than a username and password into the database—say, their own SQL queries, which would allow them to execute commands on the database. As a result, they could create new accounts with administrator privileges, change the passwords on existing accounts to let them log in to

the accounts using the new passwords, and mine the database for personal information, such as credit card numbers. The black hat can even delete the database! Figure 7-4 shows an example of a SQL injection.

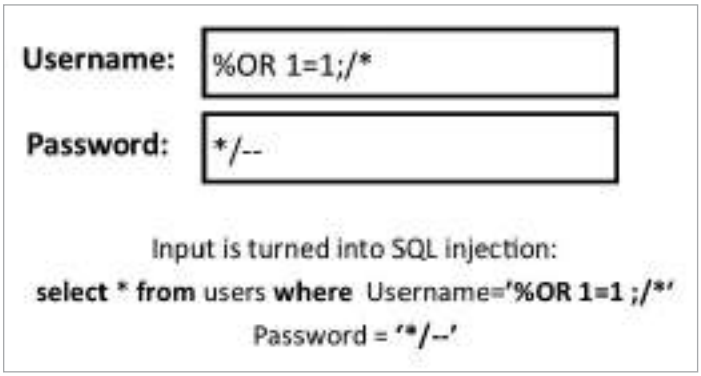


Figure 7-4: An example of a SQL attack

The top section shows how a normal SQL query is created from a username and password. The bottom section shows a standard query used to determine whether a service is susceptible to SQL injection. This query essentially tells the database to look for any user and return that information (the password field gets nullified by the */ syntax), indicating to the attacker that the service is susceptible to this type of attack.

That said, you won't often see this succeed in modern systems, because most services have recognized and mitigated the attack. As with buffer overflows, the best way to defeat this attack is to use input validation to remove malicious SQL queries before they're sent to the database. Still, it's an excellent example of how an adversary can use the backend infrastructure to attack a cloud service.

XML Injection

Extensible Markup Language (XML) is a set of rules used to create dynamic documents that web servers can read to populate web pages. For example, let's say you're on sparklekitten.net, shopping for all those cool Sparkle Kitten products. When you click an item, the web server can send an XML request to a database with price information. The database sends that information using XML, which the web server then populates into the web page, giving you the latest product price. XML is beneficial, because it works no matter what type of web server or database you're running; if both understand XML, they can communicate with each other.

Adversaries use XML's flexibility to their advantage. In an XML injection attack, the black hat creates their own XML document and sends it to the web server. The web server accepts the XML document and updates the site using the malicious input from the attacker. An XML injection

attack can behave differently depending on how the document is written. For example, it's possible to create a user with full administrator privileges using XML injection.

Often, attackers use other exploits to get the web server to validate the XML document. Like with SQL injection, they might use a field in the web page to inject the XML information into the application's backend. That means you can use input validation to defend against these attacks too. If the system can recognize the XML information, it can remove it from the input or reject it outright.

Defending the Cloud

Cloud services can be more secure than systems you own and run. The reason is that cloud providers can spend more money on security than smaller companies can. Also, because cloud providers service multiple paying clients, they can afford to maintain several security services, including hardware, software, and personnel, without having to charge any one client the full cost of those services. It's also in the best interest of the cloud provider to have a high level of security, because its entire business is based on making sure its clients can use its service securely and consistently.

However, you can't always rely on a cloud provider to have the security you need. Before signing up for a cloud service, carefully examine its terms of service to determine what level of security the cloud provider will maintain. The terms should also tell you what you, as the client, are responsible for. Many cloud providers keep compliance reports, such as a SOC II report, that you can request. These reports not only show what security features they have in place, but also provide proof that those features are functional.

One of the weakest points of cloud security is at the publicly accessible client portals. Many of the attacks discussed so far begin at the web page that the client uses to interact with the cloud service. Administrators must frequently test these portals to ensure they're secure. One way to do that is to use fuzzing, as discussed in "Buffer Overflows" on page 130. You should also add cloud applications and connections to the monitoring you're doing for the systems on your network. For example, many cloud services allow you to integrate with third-party applications, such using a Facebook login to log in to your account, or with systems that you control on your internal network, such as a customer database. Monitoring what access your cloud service has to other applications, systems, or accounts can help ensure that a black hat hasn't taken control of your cloud account or used the cloud service to gain access to your internal systems.

It's also important for cloud service users to have proper training. This preparation will help ensure that a user doesn't fall prey to a social engineering attack. Using multi-factor authentication can protect the cloud service too. Many attackers will give up and move on to easier victims rather than taking the time to break through multi-factor authentication.

Exercise: Performing SQL Injection on the Damn Vulnerable Web Application

To better understand the principles of SQL injection, let's perform the attack. We'll target the *Damn Vulnerable Web Application (DVWA),* an intentionally defenseless application designed for testing various weaknesses.

Before you can load the DVWA, you'll need a platform to run it on. Because the DVWA is, well, vulnerable, it's best to run it as a virtual machine or container. For this exercise, we'll use the service Docker to do this. Docker is a platform for running containers, which are essentially software packages designed to run regardless of where they're executed.

Installing Docker and the DVWA

To sign up for a free account and install Docker, visit *https://www.docker.com/*, as shown in Figure 7-5. Click the **Get Started** button in the upper-right corner, which should take you through the process of downloading and installing the Docker Desktop application. The standard installation will work for our purposes.

Once you've installed Docker, you can decide to go through the tutorial or get started right away.

Figure 7-5: The Docker home page

Next, you'll need to download and launch the DVWA using Docker. To do this, access the command line by going to your Start menu and entering **Command Prompt** on Windows or opening the Terminal application on macOS. Then run the following command:

```
docker run --rm -it -p 80:80 vulnerables/web-dvwa
```

This command should create the container housing the DVWA. Once the command is finished running, the output should look like this:

```
Unable to find image 'vulnerables/web-dvwa:latest' locally
```

```
Latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:
dae203fe11646a86937bf04db0079adef295f426da68a92b440e3b181f337daa7
Status: Download newer image for vulnerables/web-dvwa:latest
[+] Starting mysql…
[ ok ] Starting Maria DB database server: mysqld.
[+] Starting apache
[...] Starting Apache httpd web server: apache2AH00558: apache2: Could not
reliably determine the server's fully qualified domain name, using 172.17.0.2.
Set the 'ServerName' directive globally to suppress this message.
ok
==> /var/log/apache2/access.log <==

==> /var/log/apache2/error.log <==
```

With the DVWA successfully installed, you can open the application. Click the Docker icon in the task bar and then click **Dashboard**, as shown in Figure 7-6.

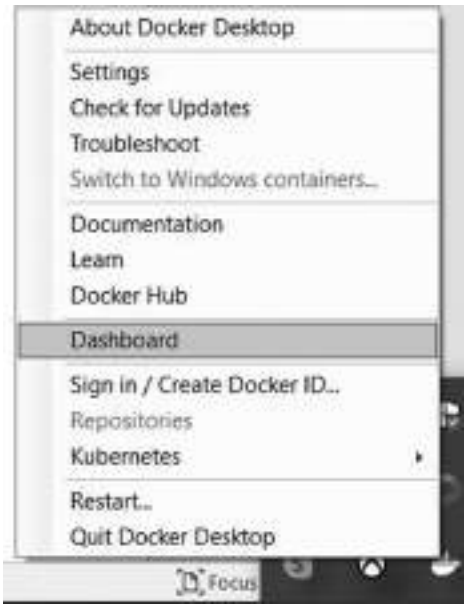


Figure 7-6: Selecting Dashboard from the Docker menu

You should see the DVWA listed as a container you can select. When you highlight the container, it should show several icons that provide different options. Click the first one, **Open in Browser**, as shown in Figure 7-7, to launch the DVWA.

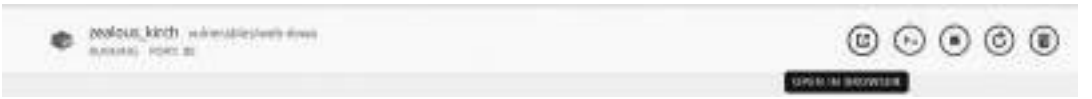


Figure 7-7: Select Open in Browser to open the DVWA container

A browser window opens to a login page for the DVWA. Use the following default credentials to log in:

```
Username: admin
Password: password
```

A screen like the one shown in Figure 7-8 appears. Here you can create a database to use for your DVWA scenarios. Click the **Create/Reset Database** button at the bottom to begin. You'll have to log in again once you do.

Figure 7-8: Creating a database for the DVWA

Now that you've finished setting up the DVWA, we can do some SQL injection.

Listing Users

Once the installation is done, various attacks should automatically be listed on the left side of the main page. Click **SQL Injection** to go to a page with a User ID field and a Submit button, as shown in Figure 7-9.

Figure 7-9: The SQL Injection field

To start the injection, let's try something simple. Enter **1** in the field and press ENTER. A red error text pop-up should appear with the following text:

```
ID: 1
First name: admin
Surname: admin
```

This error already indicates that the application is vulnerable: instead of displaying *User Not Found* or a similar message, it shows the first and last name of the user with the ID of *1*. Let's see if we can find more users. Enter the following into the User ID field and press ENTER:

```
sparklekitten' or '1'='1
```

Let's break down this input. The SQL language evaluates every query to either true or false. The `or` operator displays a record if any of the conditions separated by that operator are true. Because `sparklekitten` won't be an entry in the User ID field, this condition will be false, but because 1 always equals 1, this condition will be true. By SQL's logic, this input will essentially produce all the User ID entries in the database, because the 1=1 is true and the `or` operator refers to either condition; SQL query believes every entry meets the requirements of the query and sends every username as a result. By entering this input, you should get the following output:

```
ID: sparklekitten' or '1'='1
First name: admin
Surname: admin
ID: sparklekitten' or '1'='1
First name: Gordon
Surname: Brown
ID: sparklekitten' or '1'='1
First name: Hack
```

```
Surname: Me
ID: sparklekitten' or '1'='1
First name: Pablo
Surname: Picasso
ID: sparklekitten' or '1'='1
First name: Bob
Surname: Smith
```

As you can see, there are five users in the database.

Finding Database Table Names

Now that you know running SQL commands from the User ID input field works, you can use SQL queries to find more information about the database. For example, let's find all the names of the tables in the database. A *table* is like an Excel spreadsheet; it stores data in columns and rows. If you find the name of a table, you can make queries specific to that table.

To find a table name, use the following query:

```
sparklekitten' and 1=0 union select null, table_name from information_schema.tables #
```

This command allows us to run a SQL query that asks to select two results from the `information_schema` table: `null` and `table_name`. The `null` value will return nothing. We use `null` because we know there are two fields (first name and surname), and we have to have two results. The real focus is the table name; this query finds the table names in the `information_schema` table. In SQL databases, the *information schema* table holds the names of all the tables in the database, so we're essentially asking the database to return a list of every table. We should get a long list of names, most of which are standard tables created for running the SQL database. In the following output, only the two at the top are important for our purposes: `guestbook` and `users`.

```
ID: sparklekitten' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: guestbook
ID: sparklekitten' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: users
ID: sparklekitten' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: ALL_PLUGINS

Output abridged
```

Because we're trying to find information about users, let's focus there. Now that we know the table name for users, we can query it for more information. Try the following query to see if you find anything of interest before continuing with the exercise:

```
sparklekitten' and 1=0 union select null, concat(table_name,0x0a,column_name)
from information_schema.columns where table_name = 'users' #
```

This line asks for all the column names in the `users` table. The `concat` command asks for the table name and column name from any table named `users`. The `0x0a` syntax represents a newline so the table name and column name are printed separately, making them easier to read.

When you enter this query, the output should contain several lines showing the names of the columns in the table.

Finding Passwords

One column in particular is of interest to our goal in this exercise:

```
ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from
information_schema.columns where table_name = 'users' #
First name:
Surname: users
password
```

This output indicates that a `password` column exists. That seems promising! Use the following query to see what the database stores in that field:

```
%' and 1=0 union select null, concat(first_name,0x0a,last_
name,0x0a,user,0x0a,password) from users #
```

This query asks for the `first_name` column, the `last_name` column, the `user` column, and the `password` column in the `users` table. The output should show all of this information, giving us the password for the admin account for this database. Congratulations! You just got access to the database using SQL injection.

This scenario is designed to be easily exploitable, but performing SQL injection on other sites won't be so straightforward. Nevertheless, you should now recognize how simple coding mistakes can lead to large vulnerabilities. By shaping standard SQL queries and learning information about the names of tables in the database, you discovered a lot of useful information, including passwords. Once you have the passwords, you'll likely gain access to the database.

There are many other exploits in the DVWA. I highly encourage you to experiment with it and read the resources it provides about various attacks. By doing so, you'll learn a lot about how black hats operate.

Conclusion

When it comes to cloud services, the main factor to remember is that you're using a service that to some extent is controlled and owned by another person or group. This makes security tricky because you might not have a lot of control over how the service is secured or maintained. It also means that in general you must use the internet to access the cloud service. This opens cloud computing up to many dangerous attacks, such as web application attacks like buffer overflows or SQL injection.

To maintain your system's security, you need to research a cloud service before using it to ensure that it takes security seriously and follows best practices, such as those previously discussed in this book. You also need to carefully monitor how the cloud service is integrated with your systems to verify that a black hat isn't using the cloud service to gain access to your internal systems. Taking these precautions will go a long way toward making sure you can use cloud services with peace of mind.

8

WIRELESS NETWORK PIRATING

The wireless internet presents users with a whole new set of challenges in preventing black hat attacks. Obviously, the main difference between wireless networks and the wired networks discussed in Chapter 6 is that they're, well, wireless. Wireless networking uses radio waves to send data between devices, which means that, like the title of the 1981 hit song by Phil Collins, it's *in the air tonight*.

Joking aside, the wireless internet can be much easier for adversaries to attack because physical isolation—one of the main ways to defend against attacks on wired networks—won't work. You can potentially prevent unwanted people from accessing wired networks by, for instance, locking an office building. This doesn't work for wireless networks, because the wireless device signals travel through those barriers and can be picked up by someone on the outside.

In this chapter, you'll learn how wireless networks work and what makes them unique. After covering the underlying mechanics, we'll move on to discuss how attackers use those features to exploit wireless networks and steal data they send. Then we'll explore a few vital defenses you can use to secure your wireless network, whether at home or in an office environment. As an exercise, you'll learn how to secure a typical wireless router.

How Wireless Networks Work

Wireless networks don't use cables to transmit data; instead, the devices send wireless signals using antennas. These antennas can be external, protruding from the device like alien ears, or internal, like those likely in your laptop. Modern wireless devices have multiple antennas to increase the amount of signal they can process.

The devices on the network send and receive signals to and from a central device called a *Wireless Access Point (WAP)*, which is usually your router or switch. The WAP manages all the communication between devices by either passing the signal to another device on the wireless network or sending it to a wired network.

Wireless networks are set up in two different ways. In *infrastructure mode*, the WAP acts as both a router and a switch, serving as a gateway to the internet for the entire network. That said, there are also types of WAPs that don't have routing or advanced switching capabilities and will merely pass traffic directly between a wireless and wired network.

The other way to create a wireless network is in *ad-hoc mode*. In this configuration, there is no central WAP. Instead, each device connects directly to another device using a wireless signal. Because the devices are connected directly to one another, there is no need for a central device. Each device sends information directly to the device it's connected to. A great example of an ad-hoc network is when you connect two Bluetooth devices. Bluetooth is a form of wireless communication designed to be used over short distances, for instance, when connecting a phone to a car system to play music. When you connect a Bluetooth device to your phone or car, you're creating an ad-hoc wireless network. (This is also an example of the personal area networks discussed in Chapter 2.)

Figure 8-1 shows an example of a network diagram for each type of wireless network.

Devices can find and recognize the signals sent by different wireless networks using the WAP's unique identifier, called the *Service Set Identifier (SSID)*. Basically, this is the name of the wireless network and is what appears when you connect your device to the network, as shown in Figure 8-2. WAPs usually broadcast their SSIDs, so any device listening for a wireless signal can see them, although they can also hide their SSIDs, which forces you to know the ID to connect.

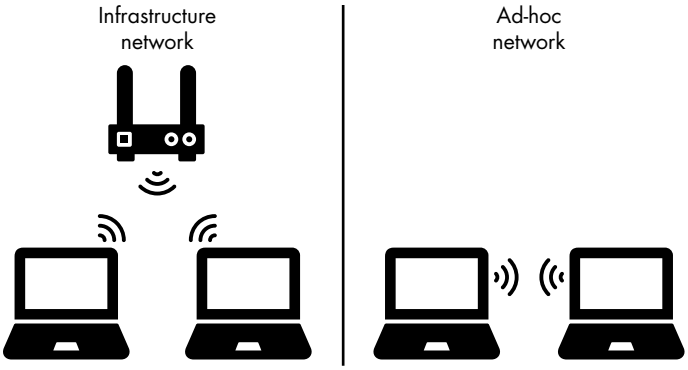


Figure 8-1: Infrastructure versus ad-hoc network setup

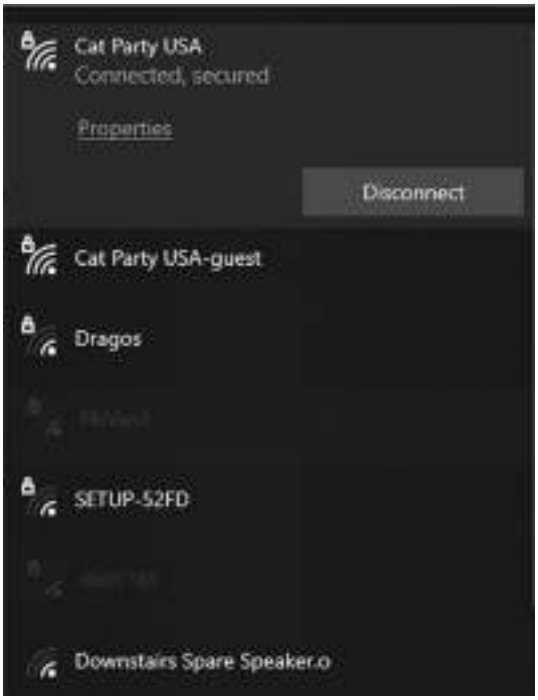


Figure 8-2: Examples of SSIDs for wireless networks

Once you choose a wireless network, your device connects to it using a process called *association*. Typically, a device like a laptop or phone can associate with only one wireless network at a time, but a WAP can associate with multiple devices simultaneously. The WAP will then manage the traffic sent to each device, making sure each one has a chance to use the signal without them interfering with each other.

Modern wireless networks can handle many connected devices. But when they reach capacity, the devices will start to experience slower speeds or lose connection to the wireless network. For this reason, it's important to consider how you'll use the network before you plug in your WAP. In a small house, you can usually get away with one access point in a centrally located area. In an office building or larger area, you'll need multiple access points to load-balance the network so no single device gets overloaded.

However, having multiple access points can sometimes cause issues, particularly when a device can't easily connect between the different networks because it can't associate with the new access point. To alleviate this problem, you can create a *mesh network*. Mesh networks combine multiple WAPs into one large wireless network with a single SSID. This allows people to move their devices freely between them without losing the wireless signal, as long as they stay within the bounds of the mesh network.

Wireless Standards

Part of what has made wireless technology so successful is the creation of widely adopted standards that enable interoperability among devices. *Interoperability* in this case means that one wireless-enabled device can easily communicate with pretty much any other wireless device, no matter which companies manufactured them. This is largely due to the work of the Wi-Fi Alliance, an organization that promotes wireless standards. Essentially, if a wireless device follows certain guidelines, it can refer to its device as *Wi-Fi enabled*.

The main standard that companies follow is known as *IEEE 802.11*. The *Institute of Electronic and Electrical Engineers (IEEE)* is an organization that produces standards, and 802.11 is just a number that categorizes that particular standard. However, that number has now become synonymous with wireless technology. In fact, it's become so common that you've probably heard or seen it before, perhaps on a router you bought for your home or office.

As wireless technology evolved, new substandards were created to govern new types of wireless networks. The first two major substandards were *802.11a* and *802.11b*, which use different radio wavelengths to send information. The 802.11a standard uses the 5 GHz band, whereas 802.11b uses the 2.4 GHz band. The 2.4 GHz band reaches farther but is slower, whereas 5 GHz is faster but doesn't reach as far. Today, these differences are largely moot, because most modern wireless devices use both standards (more on that momentarily).

Later standards integrated new features. *802.11g* improved on 802.11b, producing a faster, stronger signal. Next came *802.11n*, which added a technology called *Multiple-In Multiple-Out (MIMO)*. MIMO allows a device to use multiple antennas to amplify the amount of traffic a wireless device can send and receive at one time. This feature drastically increased a WAP's capacity, allowing for many more devices on a network at one time without causing slowdowns or dropped signals. MIMO also allows devices

to use 2.4 and 5 GHz signals at the same time. The *802.11ac* and *802.11ax* standards later improved upon MIMO and other signal-strengthening technologies to enable incredibly fast data transfer speeds. Released in 2019, 802.11ax is still being adopted. Table 8-1 lists each of the standards along with some facts about them.

Table 8-1: IEEE 802.11 Standards

Standard	Signal type	Max range	Speed
802.11a	5 GHz	120 m	54 Mbps (megabits/second)
802.11b	2.4 GHz	140 m	11 Mbps
802.11g	2.4 GHz	140 m	54 Mbps
802.11n	2.4/5 GHz	250 m	600 Mbps
802.11ac	5 GHz	120 m	3466 Mbps (3.4 Gbps)
802.11ax	2.4/5/6 GHz	120 m	9608 Mb/s (9.6 Gbps)

Wireless Security

Another interesting aspect of the IEEE standards is that they make security an integral part of their functionality. Their designers attempted to defeat some of the more obvious attacks against wireless from the beginning rather than reacting to them after the fact. Although not perfect by any means (as you'll see in "Wireless Attacks" on page 147), this emphasis on security helped create wireless security standards that have persisted through many iterations.

Wireless Authentication

One of the problems with wireless is that it can be very difficult to determine who is using the network. Because the network isn't constrained by traditional physical boundaries, such as walls or doors, it must use authentication protocols to allow only the right people to access it. The other issue is that when you send in your authentication, it travels over the wireless network, where an adversary could intercept it. So you also must encrypt your traffic to ensure your credentials remain secret.

Basically, there are two methods of authenticating to a wireless network. The first is *Personal* or *PSK (pre-shared key) mode*. This is most likely the method you're familiar with; it requires you to enter a password when you want to join the wireless network. As discussed in Chapter 5, a password is only as strong as you make it; so a weak password using Personal mode will make it easier for a black hat to break into your wireless network. However, it's an easy method to use at home, because it doesn't require any additional equipment or maintenance outside of the WAP that you set up. It's also easy to share the password to allow people to join your wireless network. This is why guest Wi-Fi networks in coffee shops or other public places use Personal mode.

The other general method of authentication for wireless networks is *Enterprise mode*, in which authentication is handled by an authentication database. The WAP forwards authentication attempts to the database to validate. The database then tells the WAP whether or not that device is authorized to be on the wireless network. These requests are sent using the *Extensible Authentication Protocol (EAP)*, which encrypts them, making it harder for an attacker to intercept them. EAP also has the added benefit of running many different types of authentication protocols, so it can integrate with many authentication methods.

One advantage of using Enterprise mode, sometimes called *802.1X*, is that it provides a way to create a unified method of authentication for wireless access and access to the network attached to the WAP. So instead of having to remember or enter multiple credentials, a person can just authenticate once to access everything they need. Another advantage is that it provides very strong authentication methods that are difficult to break. The drawback is that all these components require a lot of management and cost. Therefore, you're unlikely to see Enterprise mode used in homes or even small businesses.

Wireless Encryption

Another important facet of wireless security is ensuring that the data sent over the network isn't read by a black hat. It's possible for an adversary to capture a wireless signal, stealing any data sent between a device and the WAP. To protect the wireless network, WAPs employ special encryption algorithms.

The first of these, *Wired Equivalent Privacy (WEP)*, was introduced as part of the original 802.11 standard and attempted to provide the same level of encryption as was used on wired networks at the time. However, this algorithm didn't live up to its name. The problem was it didn't have a long enough encryption key. As you'll learn in the next chapter, if an encryption algorithm's key is too short, attackers can easily break it.

Knowing there were problems with security in WEP, the Wi-Fi Alliance helped create a new encryption standard called *Wi-Fi Protected Access (WPA)*. The idea was to create a stopgap measure to mitigate the problems with WEP while an even better standard was being worked on. WPA used a protocol called the *Temporal Key Integrity Protocol (TKIP)* to supplement the key size of WEP encryption and make it harder to break. However, flaws still existed in how the key was used, and an adversary could still break the encryption.

Eventually, another version of WPA, aptly named *WPA2*, met the goal: it provides encryption security equal to that of wired networks. WPA2 used an entirely new encryption algorithm called the Advanced Encryption Standard (AES; more on this in Chapter 9) to provide a strong level of encryption. WPA2 also included the 802.1X standard discussed earlier. This continues to be the go-to standard for wireless security. Although not flawless, as you'll see shortly, it provides a high level of encryption and authentication that prevents attackers from sniffing wireless signals or joining a wireless network without permission.

It's important to note that even though all modern WAPs support WPA2, they often also support legacy standards like WEP and WPA. For this reason, it's critical to check your WAPs to ensure they're not using a legacy standard with considerable flaws.

Wireless Attacks

The most popular attacks against wireless networks are man-in-the-middle attacks, like those discussed in Chapter 2. The reason is that wireless networks inherently make it easy for an adversary to intercept communication between the WAP and the devices connecting to the wireless network.

Other popular attacks include wireless sniffing and DoS. Wireless sniffing is largely mitigated through the use of strong encryption, but if you're using a weak algorithm like WEP, it's still a viable attack. DoS attacks are also effective because it's relatively easy to cause interference with wireless signals, sometimes even unintentionally.

Let's walk through some specific examples of attacks that adversaries use against wireless networks so you'll know how a black hat might accomplish them.

Rogue Access Points

One way for an adversary to create a man-in-the-middle situation is to imitate an access point and trick an unsuspecting person into connecting to it. This is known as a *rogue access point*, which allows the adversary to see any traffic that is sent by the victim to the fake access point. This attack is particularly easy to achieve with a wireless network because it doesn't require expensive equipment. In fact, it can usually be accomplished with open source software and a computer's wireless card.

Once the attacker turns their system into a rogue access point, anyone who connects to it will send their traffic to it as if it were a legitimate device. Even worse, the adversary will often create a bridge to a real WAP, passing the traffic to the internet and making it difficult for the victim to know that something is wrong, because they can still visit websites without any problems.

Another form of man-in-the-middle attack is called the *Evil Twin* attack. In this variation, an attacker copies the name of a legitimate wireless network to trick people into connecting to it. To do this, the adversary first finds the names of legitimate networks in the area by *wardriving*, which is a technique they use to set their wireless card to pick up all signals, even hidden ones. This gives them information about where wireless networks are located. Once they have a target, the attacker turns their device into a rogue access point with the same SSID as their target. Then they wait for someone to connect to it, thinking it's a legitimate access point. Figure 8-3 shows an example of this man-in-the-middle attack.

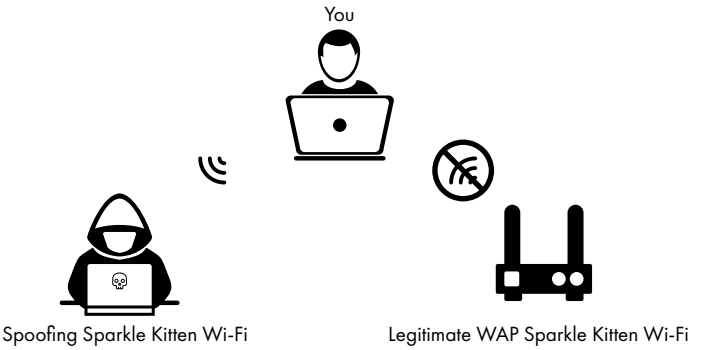


Figure 8-3: An example of an Evil Twin attack

It's fairly easy to trick users into thinking the fake access point is authentic. Because many places offer guest Wi-Fi, a black hat only has to make their network look like one of the guest networks to get people to join. This is especially easy in crowded places where many guest network options are available, such as an airport. Combine these networks with the networks that stores located throughout the terminals might offer, and users have numerous options. Effortlessly, an adversary can hide in plain sight (which is why you should never, and I really mean *never*, use a public wireless network in an airport).

The worst part of an Evil Twin attack is that many devices save a network's information so they can connect to it later. These devices might automatically connect to the Evil Twin when they pick up an SSID they remember. Essentially, when you tell your device to remember a network, it will regularly send out association frames looking for that SSID. When it hears an answer, it will connect to that WAP regardless of whether or not it's the legitimate one. So it's imperative to check which wireless network your computer is connected to and ensure it's on the correct one.

Disassociation Attacks

As mentioned earlier, connecting a device to a WAP is called association. To associate with a network, the device must send a request using a chunk of data called a *frame*. When the WAP receives this *association request frame*, it begins communicating with the device to establish a connection to the wireless network. This is also when authentication occurs, if required.

In a *disassociation attack*, a black hat reads a device's MAC address from the network, copies it, and sends a *disassociation frame* to the WAP on its behalf. The disassociation frame does (you guessed it) the opposite of an association frame; it disconnects a device from the WAP. The device must then reassociate with the access point. The attacker can perform this attack repeatedly, effectively blocking communication between the victim and the WAP.

A disassociation attack provides an adversary with a few benefits. First, it creates a DoS state for the victim. Second, it gives the adversary a chance to get the victim to connect to a rogue access point to accomplish

a man-in-the-middle attack. Third, it can be used as a means to grab authentication material to crack the password or encryption key used to secure the network. Many of the attacks against wireless encryption start by sending a large amount of association or disassociation frames to record the encrypted material that is being sent between the device and the WAP. Once the attacker has enough of this encrypted material, they can more easily use their system to brute-force the encryption key the WAP uses. This is why weak keys, like those WEP uses, are such a problem; it takes less encrypted communication to break the key, which makes it relatively fast to do.

Jamming

Another way for an adversary to attack a wireless network is to cause interference, jamming the signal so legitimate users can't connect to it. Wireless networks are especially susceptible to jamming because they rely so heavily on radio frequencies produced by many other devices, such as microwaves. So it's easy to create extra noise for jamming because many of these devices, such as telephones and wireless keyboards, will already be close to the WAP. In fact, it's easy for you to accidentally create a jamming condition. Placing devices next to powerful electronics, such as servers and large kitchen appliances, can slow down or even completely block out a wireless network.

This is especially true for 2.4 GHz networks, because many electronic devices produce that band of frequency. You can even jam a 2.4 GHz WAP by placing it too close to another WAP. The 2.4 GHz range is broken down into 12 channels. All but three of these channels overlap in some way, leaving 1, 6, and 11 as the only viable options. If two devices are on the same or overlapping channels, they'll interfere with one another, preventing users from using the WAP effectively, if at all.

Although 5 GHz networks do suffer from interference, that range has 23 nonoverlapping channels available, make it more resistant to ambient interference from other devices.

Setting Up a Wireless Network with Security in Mind

The best defense against an attack on your wireless network is to set up your WAP with security in mind. When you place your devices carefully, you can eliminate some of the opportunity a black hat might have to attack the wireless network in the first place. To do this properly, you'll need a well-organized and effective plan. One way to do this is to create a *wireless network diagram*, which is a map of the area where you'll locate your wireless network. Then add notes on where the WAP should go, how far the wireless network should extend, and what type of network you're creating.

For example, you might want to set up a wireless network that will work throughout an office building. By using a wireless diagram, you can get a general sense of how far each WAP will send its signal and how many WAPs you'll need to cover the entire building. You'll also be able to place the access points in such a way that the signal is reachable only when users are inside

the building. This will make it more challenging for an adversary to attack the network; they'll now have to bypass physical barriers, such as locked doors, to begin their wireless attack.

Wireless diagrams are also ideal for ensuring that you're not placing a WAP near devices that might cause interference. Figure 8-4 shows an example of a wireless network diagram.

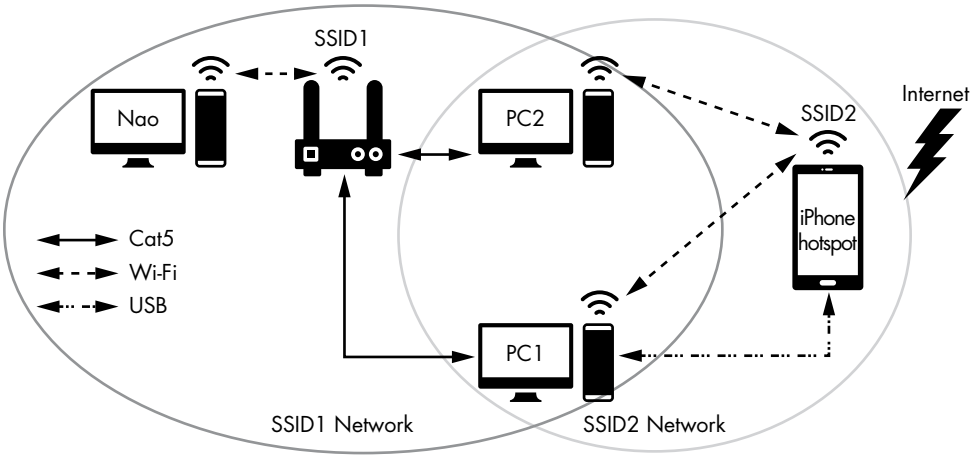


Figure 8-4: An example of a network diagram

Using a wireless network diagram, you can set up your WAPs and ensure they're configured properly. Verify that they're using WPA2 for encryption and either 802.1X authentication or a strong passphrase. Also, make sure they're segmented on their own subnetwork on your internal network. That way, any traffic traveling from the wireless subnetwork to the main internal network will have to pass through extra access controls before the traffic is allowed, similar to how DMZs work.

After you've set up the network and verified the configuration, you need to routinely move around the area that the wireless network encompasses and wardrive for any rogue access points or Evil Twin attacks. This is called a *site survey*. By sniffing your own wireless network, you'll not only be able to see whether any attackers are attempting to spoof it, but you'll also be able to find any hidden SSIDs an adversary might be using to access your network. In one famous example, an attacker posing as a janitor was switching out keyboards with identical models that included a small wireless transmitter and keylogger malware. The black hat was able to steal the keylogging data using the wireless transmitter. Although this is an extreme case, it's a perfect example of how an attacker might sneak a wireless device into your internal network to later use it for access.

The main detail to remember is that wireless networks might require even more security than wired networks because of how easy it is to pick up

a wireless signal. That doesn't mean you can't use a wireless network; you just need to be smart doing so. Whenever possible, avoid sending sensitive data over a wireless network without encrypting it first. That way, even if the wireless network encryption is broken, your data will remain encrypted.

In general, you should also avoid using public wireless networks, such as those that coffee houses and airports offer. You have no idea who might be listening in, so it's best to avoid them altogether. In addition, always make sure your wireless network is using at least WPA2 encryption or later. WPA and WEP are too easy to break to be trusted as security protocols. With these tips in mind, you can avoid black hat sniffers and enjoy the freedom of wireless access.

Exercise: Secure Your WAP

In this exercise, you'll learn how to enable your WAP's various security settings. Although certain configuration details vary based on the WAP model you own, for demonstration purposes, we'll focus on how the features work rather than any particular model's mechanisms. This approach should give you an idea of what keywords to look for in your WAP's settings and how certain tools should work. Let's start by setting up an access point.

Setting Up Your Access Point

Many access points include a setup wizard that you can use when initially setting them up. To access this wizard, you must turn on the access point and wait for it to fully boot. You should then see lights on the WAP indicating the wireless connection is running.

Most wireless access points create a default wireless network when they're first booted. Often, you'll find the name of this wireless network on the back of the access point or in the device's documentation. Once you find the network, you can connect to it, either as an open access point or by using the given credentials, which are usually printed on the device. If you don't find a default network for the device, you might have to connect to it directly with an Ethernet cable.

Once you're connected to the default wireless network or the device, you'll need to access the setup wizard. This often requires knowing the default IP address assigned to the device upon starting. Again, you'll usually find this number printed on the WAP or in its documentation. A common IP address used by default is 192.168.1.1, but there are definitely variations. After finding the IP address, enter it into your web browser and press ENTER. The browser should connect to the device and either bring up the wizard or the administrative menu to begin the setup process.

For example, Figure 8-5 shows the home page for the wizard that an ASUS wireless router uses.

Figure 8-5: A wireless access point setup wizard

The wizard should prompt you to enter a password. Ensure that you use a strong password for the administrator account, which allows you to log in and change the router's settings. Many routers include a default password for the admin account, especially if they're issued by an ISP. You should always change this default password to prevent others from using it to gain access to the settings. Even if the password is unique to that router, it's still possible for a person to discover it, because they're often printed on the router.

Next, enter an SSID, or name, for your wireless network. Sometimes the router will use a default name when you first start it. But if you uniquely name it, it will be easy to recognize among a list of other names, especially if you live in an area with multiple networks, like an apartment complex. Naming your network is also a great way to give it some personality; for instance, my network name is Cat Party USA.

Once you've set up the SSID, you can make sure the WAP's security features are configured properly.

Setting Up Wireless Security

The first security setting you'll want to check is the router's wireless encryption standard. Many routers automatically use WPA2 and might even enable it during the initial setup of the device. However, it's always best to check the standard used and that it's set up correctly to ensure the best encryption available.

To do this, you'll need to find your router's security settings. Often, they'll be listed on the wireless network configuration menu or in a

separate security configuration menu. Figure 8-6 shows an example of this menu for the ASUS router.

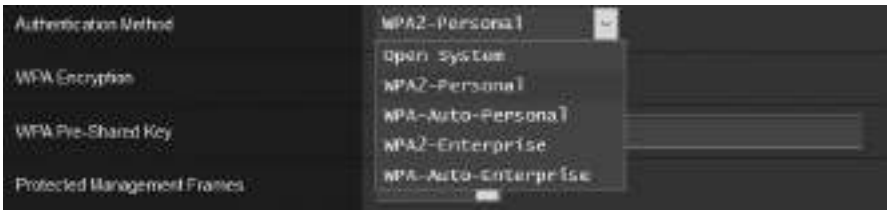


Figure 8-6: Wireless encryption options

You'll be presented with several options, the best of which for home networks is usually WPA2-Personal. It allows you to set up a *pre-shared key*, which is essentially a password you'll use to join the network. Remember to use strict password best practices, as discussed in Chapter 5. Because others will be able to access the login menu, make sure the password is strong enough to resist brute-force attacks or other password-cracking techniques.

You can also choose the encryption standard to use with your network. We'll discuss encryption in more detail in the next chapter; for now, know that the best option to select is *AES*, which is currently the most effective standard and provides extremely strong encryption.

Next, let's look at a few other security features you need to configure. The first is *Wi-Fi Protected Setup (WPS)*. This feature allows you to access a WAP without having to enter long passwords or do other complicated security setup. The idea is that to log in to the network, you use a PIN code or press a physical button located on the WAP. However, WPS is flawed, and it's possible to brute-force the PIN code to gain access to the network. In general, it should be disabled. You can usually find the configuration settings for WPS in the wireless network settings. Figure 8-7 shows an example of what this setting might look like.

Figure 8-7: The WPS setting

You should also look at your WAP's remote access settings. It's possible that someone can access your WAP's administrative side from the internet without having to be on the network. This is dangerous, because it can lead to an adversary gaining access to the device without having to be physically near it. Check your settings to make sure nobody can access the WAP from the WAN (aka the internet). Figure 8-8 shows an example of the remote access settings.

Figure 8-8: Remote access settings for an ASUS WAP

Another remote setting to check is how the connection to the access point settings is secured. Avoid using Telnet or HTTP. Neither provide encryption, meaning it's possible for an adversary to see what you're doing whenever you modify the settings on the router. Instead, use SSH or HTTPS. SSH requires more setup, because you must create a special encryption key for your device; the easiest option is to use HTTPS. This will secure the connection to the WAP when you use your web browser to access the administrative settings.

Enabling Filtering

Filtering allows you to selectively accept connections based on parameters of your choosing, and then block all other devices or traffic. Depending on your router model, you might have several options for filtering. Let's look at some of the more common options available.

One option, *port filtering*, allows you to filter traffic based on the port number associated with it. Using port filtering is a great way to limit which protocols your network allows. For example, you might block port 21 because it's associated with FTP, which is an unencrypted way to move files over a network. You might also want to block port 23, which Telnet uses; it's also an unsecure remote access protocol.

Regardless of what you want to block, be careful. As with all filtering, it's possible that blocking a port will cause unintended consequences. For example, you could accidentally block an application that you're using. Figure 8-9 shows an example of port filtering settings. In this example, the WAP allows you to set certain times at which to block ports and displays whether you're specifying a whitelist or blacklist.

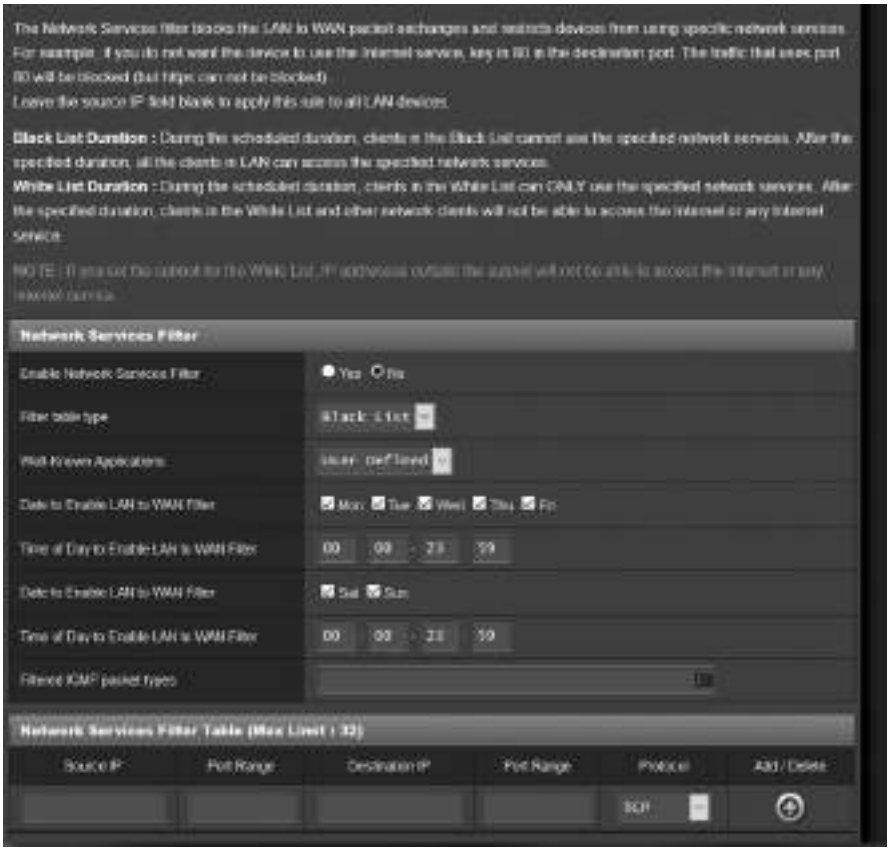


Figure 8-9: Port filtering settings for an ASUS router

Another form of filtering, *URL filtering*, allows you to block certain URLs, or even phrases used in the URLs, from being resolved by the WAP. Basically, when someone tries to access one of these URLs through the WAP, the WAP will recognize the website address and block it. This is a great way to create parental controls or limit unwanted content from being accessible on your wireless network. However, URL filtering requires that you be very specific, and it can often filter legitimate websites by accident. As with port filtering, it's best to test it to make sure it's working properly.

Another common filtering option for WAPs is *MAC address filtering*. As discussed in Chapter 6, switches use MAC addresses to determine where they should send traffic on a LAN. MAC filtering lets you filter access to the wireless network by device, using either a blacklist or a whitelist. Typically, a whitelist is most effective, because you're more likely to know the MAC addresses of the devices you *want* on the network than those of the devices you *don't* want. If a device tries to connect to the wireless network without being on the whitelist, the WAP will reject it.

Although MAC filtering is an excellent way to limit who can access a network, it's not the best at stopping black hats. Because MAC addresses are

sent unencrypted, an attacker can easily sniff wireless networks and collect the MAC address of a device that is allowed on the network. Once they have an address on the whitelist, they can easily spoof it to gain access. Typically, it doesn't matter to MAC filters if two devices have the same MAC address as long as that MAC address is on the list. This is a good example of why it's important to layer your network's protections: although MAC filtering might not be effective on its own, it adds to a device's overall security.

Figure 8-10 shows an example of MAC filtering settings.

Figure 8-10: MAC filtering settings on an ASUS router

With these settings in place, your WAP will be secure and guard against attackers and unwanted actions by legitimate users. Although configuring all these setting might seem like a lot of work, it's always better to be safe than sorry. You don't have to use every feature mentioned to secure your network, but the more layers of security you have in place, the less likely it is that a malicious attack will occur.

Conclusion

Wireless networks present a unique security challenge. Although convenient, a network's capability to span a large area makes it easy for attackers to access. It's also possible for an attacker to trick people into using a wireless network that they control. This is why it's important to use only wireless networks you trust and avoid using public wireless networks. For your own wireless networks, using encryption (such as WPA2) and authentication (like 802.1X) can help prevent attackers from compromising them, allowing you to use the internet with more confidence in your network's security.

9

ENCRYPTION CRACKING

Imagine if all your internet emails, messages, and purchases were available for everyone to see. A black hat would just need to watch your network traffic to learn anything they wanted to know about your internet browsing, steal your files, or even capture passwords. One of the most important parts of cybersecurity is *cryptography*, or the process of hiding data so only authorized people can see it.

In this chapter, you'll learn how encryption works, how attackers try to break it, and how you can strengthen it to protect your personal information from prying eyes.

What Is Cryptography?

Cryptography is the study and art of writing codes. When people want to hide data, they often do so by writing a code or process, known as a *cipher*, to conceal the information. In modern computer systems we hide data using *cryptographic algorithms*, which are procedures that perform complex math to securely transform information.

Ciphers and algorithms work in various ways, but they all use keys to function. A *key* is a piece of unique information that usually remains secret. Using the key, a person can turn *plaintext*, or data that hasn't been encrypted, into *ciphertext,* or encrypted data. The encryption process is similar to putting a lock on a door. When a door is unlocked, anyone can access what's on the other side. Locks, like cryptography, restrict access to only those who have the key. Every door has a unique lock and key that can be used to keep whatever is on the other side of the door safe. Encoding data using a key is known as *encryption*. You can think of this like locking a door. Reversing the encryption, a process equivalent to unlocking the door, is called *decryption*.

MEET ALICE AND BOB

To make cryptography easier to understand, experts in the field devised a narrative using two characters known as Alice and Bob. Alice and Bob represent not only *people* communicating securely using encryption, but also *computer systems* that use encryption to send information securely. Computers can do this automatically. In fact, much of encryption is handled by your computer in the background without you having to do anything or even know that it's happening. So, when you read about Alice and Bob in this chapter, just remember that encryption is not always a process in which people consciously engage.

What We Encrypt

We use encryption on many types of files and processes. Basically, anything stored on a computer can be encrypted in some form or fashion. However, the type of encryption used might change depending on the state of the data. All data is organized into one of three categories: data at rest, data in transit, and data in use. *Data at rest* includes any file that a computer isn't actively processing; for example, music, documents, or information in a database. *Data in transit* is any data being sent between two systems, including packets (as discussed in Chapter 6), emails, or files in the process of being sent over a network. *Data in use* is data that a computer system is currently using. This type of data can't be encrypted, because the computer must be able to read it to use it.

As data moves between different states, the encryption type changes. For example, imagine you install a game on your computer. If you're not playing that game, you can encrypt it using *file encryption*, storing it on your hard drive or other storage medium. But as soon as you open the game, it becomes data in use and must be decrypted by the computer to function. Now let's say you send the game to a friend over the internet. At that point, the game becomes data in transit and must now use *transport encryption*.

The different options for encryption help distinguish the methods used to encrypt that particular type of data. Each encryption option has its strengths and weaknesses, which we'll discuss in more detail in "Modern Cryptography" on page 160. For now, keep in mind that just because you're using one type of encryption, such as file encryption, that doesn't always mean your data is safe in other states, such as when it's in use or in transit.

Early Cryptography

The practice of writing encrypted messages has been around for as long as writing has. Militaries used early cryptographic techniques to transport messages, ensuring that the enemy couldn't learn about their plans should the messages fall into the wrong hands.

Early cryptography was also fairly ad hoc. People had no set standards for how to encrypt messages, so they typically invented strategies on a case-by-case basis. At times, the lack of standards presented problems when transporting messages to a stranger, such as a king or general in another kingdom, because they didn't know the encryption method. In fact, some historical writings have yet to be interpreted because experts believe they're written in long-lost ciphers.

The absence of standards highlights one of the primary problems in cryptography: how do you create a system that hides information well while still sharing a decryption key between two people? In other words, this is a problem of balancing the CIA triad elements of confidentiality and availability. If the encryption system is too difficult, it can be challenging to use effectively. For example, if I encrypt a message so it can only be decrypted on the highest peak in a kingdom during a full moon while saying magic words backward in Elvish, I might very well prevent enemies from reading it. But if I want to share my work with other wizards, it will be problematic for them to read it too. However, if I create a cipher that uses a keyword for decryption and then write that word at the top of the message, the encryption becomes pointless because anyone who gets the message will know the keyword as well.

Balancing availability and confidentiality considerations is an ongoing problem in cryptography. To help solve this problem, cryptographers began relying on two methods of encryption: *substitution* and *transposition*.

Substitution Ciphers

To create a substitution cipher, you replace one symbol with another. For example, consider the fictitious Super Awesome Substitution Cipher

in which you replace each letter of a message with a number. To use the Super Awesome Substitution Cipher, you would only need to know one of the substitutions, which could be used as the key. That key might be *A is equal to 10*. Once you know the key, it's easy to continue counting, pairing each subsequent letter with the next number: *B* becomes *11*, *C* becomes *12*, and so on, ending with *Z* becoming *36*. Using this cipher and key, the message *attack now* becomes *10, 29, 29, 10, 12, 20, 23, 24, 32*.

This substitution cipher wouldn't be very secure for practical use, because it's simple to crack through trial and error. However, throughout history, many effective ciphers have used the substitution method. The most famous is the Caesar Cipher, which Julius Caesar invented. It relied on the use of two copies of the alphabet. The writer lined up the two alphabets, one on top of another, so they matched: the letter *A* in the top alphabet aligned with the letter *A* in the bottom alphabet. The writer then shifted the bottom alphabet left or right by a few spaces and used the shifted alphabet to encrypt their message. The number of spaces shifted was the key. For example, to encrypt the letter *A* with a key of 3, you would write it as the letter *D*. Although it's still easy to break, this was the first cipher to create a standard substitution algorithm, and it became the basis for future substitution ciphers.

Transposition Ciphers

Whereas a substitution cipher replaces a message with completely new symbols, a transposition cipher uses the same symbols but arranges them in a different order to hide their meaning. This process resembles the word jumbles you see in newspapers or online. When put through a transposition cipher, the message *attack now* might become *kcnwta toa*, for example.

For a transposition cipher to be an effective form of encryption instead of a fun brainteaser, you must scramble the letters in an organized fashion. There are several ways to accomplish this, but one of the oldest is the *Scytale* method. Invented in Greece, the original Scytale method used a piece of cloth that was wrapped around a rod of a certain thickness and length. Writers wrote the message across the cloth, and when they unwound it the message was naturally scrambled along its length. A person had to wrap the cloth around the same type of rod to decrypt the message. To conceal the messages even further, people sometimes wore the cloths as belts.

Modern Cryptography

In the 1800s, governments began to use more systematic approaches to cryptography. This included building machines to handle the encryption/decryption process, a method of cryptography best illustrated by the German Enigma machine. Used during World War II, the Enigma used three rotating dials that encrypted and decrypted messages being sent via radio. Theoretically, the cipher was too complicated to break without an Enigma machine to do the work, because the machine's use of three unique disks generated an extensive number of symbol combinations.

But the cipher wasn't unbreakable. The universal flaw in cryptography is that any cipher can be broken, given enough time, even if it takes a billion years to try every possible key combination. The real value of a cipher is its *work factor*, meaning that it takes long enough to break that the message is no longer relevant. By the end of World War II, *cryptanalysis,* or the study of breaking ciphers, was so advanced that researchers could break any code that relied on a mechanical process, such as the one the Enigma machine used, in a short amount of time. This lowered the work factor of these ciphers, essentially making them worthless.

The advent of computers further reduced work factors, but also created new means of creating ciphers. Computers made it effortless to perform advanced math calculations quickly. As a result, people were able to solve problems in a matter of minutes that would normally have taken them days or weeks to finish. Starting in the 1950s, modern cryptographers began working on new methods of encryption that used exceptionally complicated mathematical problems to compute the numbers necessary to create the ciphertext. For example, using two specific points on an elliptical curve graph that are paired together, you can create a set of keys to encrypt and decrypt ciphertext. Even if you knew one of the points, it would take a high-end computer many, many years to search through all the possible pairs for the correct one.

Eventually, cryptographers created three main methods of encryption: *symmetric*, *asymmetric*, and *hashing*. These three methods became the foundation of modern cryptography and are what most encryption is built upon today.

Symmetric Cryptography

Symmetric cryptography, also known as *single* or *private key cryptography*, uses one key to encrypt and decrypt plaintext (Figure 9-1).

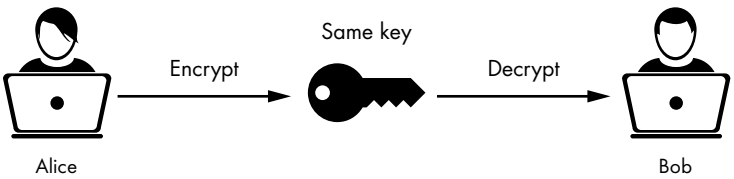


Figure 9-1: Encrypting with private key cryptography

Symmetric key algorithms use complex functions to create ciphertext from plaintext, but at their core they still use substitution and transposition techniques. Instead of switching and moving letters, they move around the computer's 1s and 0s that represent those letters. They're much stronger than traditional transposition and substitution techniques, because unlike a human manipulating data by hand, a computer can perform hundreds, if not thousands, of transformations in a short amount of time. Computers can also change how they perform the substitution or transposition while

running their algorithm. However, this can be tricky for two reasons: the system needs to scramble the 1s and 0s to ensure they can't be read again, but also make sure the process can be reversed given the correct key.

To make these processes easier, symmetric key algorithms use two types of cipher modes: *stream* and *block*. In stream cipher mode, the computer encrypts one bit of data at a time. This mode creates some of the fastest encryption but is generally considered weaker than block modes, because it generates less randomness in the resulting ciphertext; in other words, it produces patterns that can make breaking the algorithm easier, especially if the key is too short or if the same key is used frequently.

Block ciphers encrypt fixed-length blocks of bits. This technique is slower but stronger than using a stream cipher. Instead of encrypting each bit one at a time, the algorithm organizes bits into blocks. For example, a 4×4–sized block of bits is taken from the data you're encrypting, and an equally large block of bits is taken from the key. Then the algorithm encrypts the whole block of bits at once, creating an encrypted 4×4–sized block of bits.

Because the blocks are all the same size, the algorithm can combine a previously encrypted block with the key to create a new, unique input to use for the next block that is encrypted. Therefore, not only does the key help create a unique encryption, but what is being encrypted also helps increase the uniqueness of the final ciphertext, making it incredibly hard to break.

Just like locking a door with a key, symmetric algorithms are fast when encrypting files, making them efficient at bulk encryption, such as encrypting numerous files on a hard drive or an entire database. But this speed comes with a catch. If two people attempt to communicate using a symmetric algorithm, they must find a way to transmit the key without exposing it. This requires finding some method of transportation other than the one used to send the ciphertext, because anyone who intercepted the ciphertext would also intercept the key if they were together. The type of transfer needed is known as *out of band* transportation and can be tricky to do in many situations.

Many symmetric algorithms are available, but two in particular are well known. One is the *Data Encryption Standard (DES)*, which is one of the first modern encryption standards widely used among government, military, and public systems. This algorithm implements a block cipher with a 56-bit key. Although 56 bits resulted in an effective work factor when DES was invented, since the early 2000s it's been easy to break a DES key in less than 24 hours due to an increase in computer processing power. The more bits there are in a key, the more combinations a computer must go through to guess the right key. But as processors became faster, it became possible for computers to calculate the possible combinations more rapidly. A 56-bit key just doesn't produce enough combinations to prevent modern processors from guessing the key quickly.

To get more life out of DES, cryptographers invented a new method of using the algorithm, called *3DES* (pronounced "triple DES"), which, as you might guess, used three DES keys instead of just one. However, this method

included a flaw in how the encryption was executed. Black hats can exploit this flaw to break 3DES nearly as quickly as DES.

Because DES was at the end of its life cycle, people needed a new algorithm to replace it. Many cryptographers tried to develop a replacement, and in the end, they settled on the Rijndael algorithm. Today, we commonly call the implementation of this algorithm the *Advanced Encryption Standard (AES)*. AES typically uses a key size of 128 bits, although the key can be as long as 256 bits if needed. AES uses the block encryption method. Although first implemented in 2001, it's still very much in use today and has an excellent work factor. Even with current supercomputers, it would take a billion billion years to brute-force an AES key—longer than the age of the universe.

Asymmetric Cryptography

Although symmetric cryptography works well for most encryption needs, such as protecting files or creating a secure connection between two devices, the inability to easily share keys between two parties makes it problematic to use effectively to communicate across the internet. This is where asymmetric cryptography, also known as *public key encryption*, steps in.

In asymmetric cryptography, the algorithm uses two keys for each party: a public key and a private key. The keys are linked, so anything that the public key encrypts only the private key can decrypt, and vice versa. No one key can encrypt and decrypt the same ciphertext. This system is useful because any data encrypted with the public key can only be decrypted with the private key, so you can share the public key in the same message as the data you intend to encrypt (Figure 9-2).

Figure 9-2: Encrypting using a public/private key pair

For this reason, asymmetric cryptography makes it easy to encrypt communication between two people, especially when they don't know each other. For example, let's say Alice wants to send Bob a private document using email. Before she encrypts the document, she needs to be assured that Bob can decrypt it once it reaches his inbox. If she were using a symmetric key, she wouldn't be able to send the key with the email, because anyone who might be sniffing the traffic would see the document and the key. She could put the symmetric key on a zip drive and send it to Bob in the mail, but this would take time and effort. Instead of using a symmetric key, Alice could ask Bob to send her his public key. Bob can send this key over the internet by any convenient means because it doesn't have to remain a secret. Alice can then encrypt the document using Bob's public key and send the document back

by email. If an adversary intercepts the email, they won't be able to decrypt it, because only Bob's private key can do that. If his private key is secret, it doesn't matter who has his public key.

Public key cryptography is also beneficial for proving that someone is who they say they are and to keep them from refuting that they sent something, a concept called *non-repudiation*. Let's say that Bob reviews the document, says it looks good, and sends it back to Alice. Alice wants to confirm that Bob sent the document instead of someone pretending to be Bob. To do this using asymmetric cryptography, Bob can encrypt a hash of the document using his private key and send that to Alice (we'll discuss hashes in "Hashing" on page 166). Only Bob's public key can decrypt the file. If the decryption works, Alice knows that Bob sent it, because he is the only one with access to the private key. In addition, Bob can't later say that he didn't send the document, because the key pair is unique to him. Unless Bob's private key is compromised, no other person could have used it to encrypt the information.

Much like in symmetric cryptography, many different algorithms use asymmetric keys. However, these algorithms don't use block or stream ciphers. Instead, they rely on extremely complex math problems to create their key pairs. For example, one of the first asymmetric algorithms worked by calculating discrete logarithms. Another uses the factoring of large prime numbers. Although the details of these math problems are outside the scope of this book, the keys are the numbers required to solve the problems. The problems are so complex that, without the numbers, it would take a very long time for a computer to calculate the values necessary to solve them. This is what creates the algorithm's work factor.

For a long period of time, the de facto standard for asymmetric cryptography was an algorithm known as *Rivest–Shamir–Adleman (RSA)*. Named after its inventors, RSA was the most widely used public key cryptography algorithm on the internet from 2000 to 2013. Its main key size was 1024 bits, which is much larger than the keys symmetric cryptography uses. In recent years RSA has fallen out of favor, because in 2013 it was proven that the US government could crack 1024-bit RSA keys using supercomputers. Since then, most systems use either a 2048-bit key, which is substantially more complex, or another algorithm.

One of the more popular replacements for RSA is *Elliptical Curve Cryptography (ECC)*. Although asymmetric cryptography is much slower than symmetric cryptography and creates substantially larger files, ECC is one of the fastest asymmetric algorithms. It can encrypt files 10 times faster than RSA. ECC also doesn't require nearly as much processor power to use. Its typical key size is 160 bits, but it can also use 256-bit keys. As a result, we can implement ECC on devices that don't have a lot of memory, such as IoT devices.

Validating Public Keys

After reading about public key cryptography, you might be wondering how Alice initially gets Bob's public key. This is a bigger problem than you might

first expect, because although it doesn't matter who receives that public key, you must make sure that you're using the correct public key for the message's recipient. Doing so is especially important when communicating across the internet, because you have no idea whether the recipient is the person you think they are. They might be a black hat pretending to be someone else. The two main ways to verify public keys are *webs of trust* and *certificate authorities*.

Webs of Trust

A web of trust is a network of trusted individuals in which each person vouches for the people to whom they're connected in the web. For example, Alice knows Bob in real life and trusts him, so she creates a trusted link between her and Bob, and her system accepts Bob's public key. Bob not only trusts Alice but also Charlie. Therefore, Alice also trusts Charlie, because her trust of Bob transfers to every person Bob trusts (which also includes Danielle, Eric, and Frankie in the web).

Although the web of trust works well for small organizations that don't want a centrally managed trust system, it doesn't work well on the internet. With the billions of people using the internet, it would be nearly impossible to make sure that every person in the web of trust was trustworthy. Adversaries could easily exploit the web to trick people into sending them sensitive information. Therefore, only a few applications use the web of trust system. One of the most famous is called *Pretty Good Privacy (PGP)*. PGP is an encryption system primarily used to encrypt email. It was created as an alternative to centralized trust solutions that might become compromised or snooped on by government or private organizations.

Certificate Authorities

The vastly more popular way to verify that public keys are authentic is to use a *digital certificate*. Similar to a high school or college diploma, a digital certificate is a document that verifies that the public key you received truly did come from the person who sent it. To maintain and manage a digital certificate, a person uses the services of a *certificate authority (CA)*, a third-party organization that creates, manages, and validates certificates for individuals or organizations. Essentially, it tells you whether a public key can be trusted.

CAs provide several different services. One is that they create certificates for people or organizations. When an organization, for example Sparkle Kitten Inc., wants to use public key encryption for its website, it chooses a CA to provide the company with a certificate. That CA then verifies that Sparkle Kitten Inc. is a legitimate company and that the person requesting the certificate in fact represents Sparkle Kitten Inc. Once they've done so and received a fee, the CA creates the digital certificate and provides a copy to the company. The CA also creates the public/private key pair that accompanies the certificate. Typically, these certificates expire in a year, at which point anyone who has the certificate will receive a warning. Be mindful of expiring certificates when you're visiting a site. Even though the owner might just have forgotten to renew their certificate, black hats can use expired certificates to trick you into connecting to an unsafe website.

Another service CAs provide is key management and recovery. Let's say that Sparkle Kitten Inc. encrypts its entire database using a symmetric key and then encrypts that symmetric key using its CA-issued public key (a common practice for sharing symmetric keys across the internet). The private key would be the only way to access the symmetric key and decrypt the database. Now, let's say that one of the resident company cats knocks the system that stores the private key out a window, destroying the hard drive so no data can be recovered from it. Without the key, the data encrypted on the database would be lost forever. Fortunately, CAs can retain copies of the public/private key pairs they create in highly secure vaults for just such emergencies, if they provide this service and you pay an additional charge for it. Once Sparkle Kitten Inc. provides the authentication materials needed to prove that it legitimately owns the key, the CA releases the private key back to the company, allowing it to recover the database.

Additionally, CAs provide a highly managed repository of all the publicly available certificates that they manage. This allows anyone on the internet to validate that the public key sent to them is from the right person. For example, when a customer goes to Sparkle Kitten Inc.'s website, they will receive a digital certificate and a public key to use to secure the connection. The customer's browser will then contact the CA that issued the certificate to verify that the certificate came from that specific CA. If the CA verifies it as authentic, the browser can trust the public key that was sent to it. CAs maintain lists of bad certificates and routinely update their lists so adversaries can't use compromised or outdated certificates.

Hashing

One of the most widely used forms of cryptography is *hashes*, which are one-way cryptographic functions that always provide the same encrypted output given the same input. Once data is encrypted using a hashing algorithm, it can never be decrypted. For example, if I put the word *CAT* through a hashing algorithm, the output might be `x5d7nt2k`. Every time I put the word CAT into the algorithm, it will come out the same. But if I change even one letter, say creating the word *PAT*, the entire ciphertext will change to a completely different output; for example, `l3loi2jd`. Even changing a letter's case, say from a capital *C* to a lowercase *c*, would substantially change the ciphertext. This is known as the *waterfall effect*, and it's one of the main purposes of hashing algorithm; that is, making sure that even the most minor change creates a vastly different ciphertext.

It might be difficult at first to understand why we need an algorithm that always creates the same output and can't be reversed. One of the reasons hashing is useful is that it provides a way to verify that information is authentic without revealing the information. For example, we use hashing to verify passwords. Rather than storing passwords in plaintext—an incredibly unsafe practice that makes them easy to compromise—applications usually put passwords through a hashing algorithm and then store the hash. When users want to log in to their accounts, they enter their passwords, which get put through the same type of hashing algorithm. Instead

of sending the password in plaintext, the application sends the hash over the network. When it reaches the system, it compares that hash to the one stored in its database. Because a hashing algorithm's output never changes if the input is the same, the system knows that the user entered the correct password if the hashes match. Also, this means the user is the only one who knows the value of the password in plaintext form.

We also use hashing to verify files. I can run a file, such as a piece of software, through a hashing algorithm to get a hash and then distribute that hash to the public. When someone downloads my software, they can run it through the same hashing algorithm and compare it to the hash I provided. If both are the same, they know that it's the original software and hasn't been modified by another person, perhaps by adding malicious code like a virus.

Many hashing algorithms are available for public use. The first algorithm to gain popularity was Message Digest 5 (MD5). MD5 had a *digest size* of 128 bits, which means that its output was always 128 bits long, no matter how much data was input. Although this might seem like a lot of bits, it's not nearly enough to prevent an attacker from cracking the hash (more on this topic in "How Black Hats Break Hashes" on page 171). Due to MD5's flaws, security professionals replaced it with Secure Hashing Algorithm 1 (SHA-1). SHA-1 has a digest size of 160 bits, which, although better than MD5, is still too small to survive modern hacking techniques. Currently, we use either the SHA-2 or SHA-3 algorithms. Both have much larger digest sizes, typically 256 or 512 bits.

It's also important to note that users usually don't have to do anything to use a hashing algorithm. Hashing occurs as part of normal communication between your computer and an application or server. In fact, your computer doesn't even choose which algorithm it uses; instead, the service to which it's connecting decides which one it should use.

What Happens When You Visit a Website?

Now that you've learned about the different types of modern cryptography, let's revisit what we know about encrypted traffic flowing across the internet. Let's say you want to visit a secure website, such as sparklekitten.net. First, the Sparkle Kitten web server sends you a digital certificate. This certificate includes three pieces of important information: the public key for that web server, the type of symmetric key algorithm it will accept, and the CA that created the certificate. Then your system verifies that the certificate is legitimate by contacting the CA (or checks whether it's signed by an already trusted CA, such as a trusted root authority).

If the CA validates the certificate, your system creates a symmetric key using the algorithm the web server dictates. Because public key encryption is so slow, you must use a symmetric key to send data efficiently to sparklekitten.net. However, to be secure, sparklekitten.net can't directly send you a symmetric key, because any black hat intercepting traffic from the web server could capture it and break the encryption. Instead, you create a symmetric key and use sparklekitten.net's public key to encrypt it. Now

only sparklekitten.net's private key can decrypt the symmetric key. You also send a hash of all the data you just sent to sparklekitten.net to use as an integrity check.

Once sparklekitten.net's web server receives the encrypted symmetric key, it decrypts the key using its private key. Then the web server uses the hash to verify the key's integrity. If everything looks good, the web server confirms the connection, allowing you and sparklekitten.net to communicate securely using the same symmetric key. This entire process usually takes less than a second to complete. Figure 9-3 provides a breakdown of how this communication works.

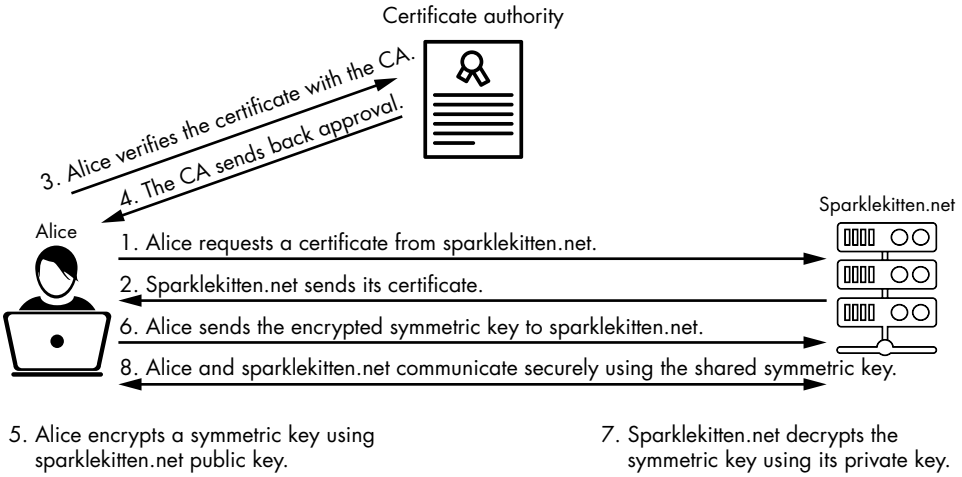


Figure 9-3: Creating a secure connection with a website

Although we use hashing and symmetric and asymmetric cryptography in a variety of ways, securing communications, as in the preceding website example, is one of the more common uses. The main aspect to remember is that systems usually handle keys and encryption in the background. As a user, you don't need to do anything to secure your system using the power of encryption. In fact, you might not even realize it's happening, making encryption simple, fast, and effective for all users. However, this lack of awareness can lead to attacks by adversaries you might not even know about.

How Black Hats Steal Your Keys

Circumventing encryption is a central part of being a black hat. An attacker can work around the security encryption provides in many ways, most of which involve another type of attack, such as social engineering, which renders the presence of encryption irrelevant. For example, you might have full-disk encryption on your hard drive, but if an adversary tricks you into revealing the pin code or password you use to access the drive, it doesn't matter how strong the encryption is. This is also true if they can trick you into negotiating an encrypted connection with them

instead of with a legitimate service, as is the case in many man-in-the-middle attacks. If this situation occurs, you share the symmetric key you generate with the adversary, allowing them to decrypt your traffic before passing it on to the legitimate service.

Attackers can also steal keys stored on hardware, especially those that applications use. Keys stored on hardware are generally harder to compromise because the attacker usually has to physically interact with the system to steal those keys. For example, your keys might be stored on a USB drive that needs to be plugged in for them to access those keys. If the adversary doesn't have the USB drive, they can't get the keys. Certain vulnerabilities, such as errors in memory storage or problems with how CPUs access information, have made it possible to access keys stored on hardware, although these are normally very rare and difficult to exploit.

These attacks against hardware don't attack the encryption algorithm but instead affect its implementation. Many cryptography compromises are caused by flaws in how the encryption ran at the time of use rather than how the algorithm is supposed to run in a perfect situation. One of the most famous examples is the wireless encryption standard WEP. It used the RC4 algorithm but chose an extremely small key space and reused information to create keys. This led to a compromise of the standard, even though the underlying algorithm was strong.

Cryptanalysis

The study of cryptography and how to break it is known as *cryptanalysis*. Modern cryptanalysis searches the inner workings of an encryption algorithm for any possible flaw that might reveal the plaintext of an encrypted message. This includes not just methods for finding the key, but also ways to decode the plaintext without revealing the key.

Researchers use a variety of techniques during cryptanalysis. One such technique is brute-force analysis. As mentioned earlier, any algorithm can be broken given enough time. The problem is, if brute-forcing the algorithm takes a long time—say, years or decades—the information obtained upon breaking it might no longer be useful. Therefore, brute-force analysis also includes methods to reduce the time it takes to correctly guess a key by exploiting flaws in how algorithms process information. For example, 3DES's main flaw was that, at its core, it used three different DES keys during its encryption process. By analyzing the algorithm, researchers found they could isolate each key separately, essentially turning them into normal 56-bit DES keys, which could then be broken using traditional brute-force techniques, thus neutralizing the strength of 3DES. This attack became known as *meet-in-the-middle* (not to be confused with man-in-the-middle).

Another cryptanalysis technique is analyzing how plaintext is converted to ciphertext. One of the more common ways of doing this is by using *differential analysis*, which focuses on differences in plaintext inputs and ciphertext outputs. This method creates various plaintext inputs to a cipher, all identical except for a controlled variable, such as a varying number in each input. The researcher then analyzes the ciphertext output and looks for statistical

patterns based on the plaintexts used. Another analysis technique is *integral analysis*, which considers a block cipher's substitution method, looking for statistical patterns in how a cipher manages its substitution. Both types of analysis can often discover exploitable flaws, such as the meet-in-the-middle flaw mentioned previously.

Cryptanalysis doesn't just focus on how the algorithm encrypts plaintext; it also looks at how it interacts with the systems around it. A *side-channel attack* focuses on elements that are outside the algorithm, such as its power consumption, timing, and even any sounds the encryption process generates that could potentially lead to the discovery of critical flaws.

Asymmetric Algorithm Attacks

Although many cryptanalysis techniques work equally well on symmetric and asymmetric algorithms, the nature of public key cryptography makes it more vulnerable to brute-force attacks. Because asymmetric algorithms rely on complex mathematics to create their key pairs, adversaries could potentially use any method that helps solve a problem faster to break an asymmetric algorithm. For example, your algorithm might rely on the difficulty of calculating discrete logarithms, creating keys from the numbers required to calculate the discrete logarithm between a pair of numbers chosen at random. If a mathematician finds a faster way to do this, they might significantly reduce the work factor of your algorithm.

The security of RSA was diminished when Edward Snowden, an NSA contractor, revealed that supercomputers could accomplish the factoring of prime numbers required to create the public/private key pair. This meant that the typical key length of 1024 bits was no longer viable, and the algorithm had to use 2048-bit keys to be secure. As computer processors continue to increase in power, asymmetric cryptography must constantly update its algorithms and standards to avoid being broken.

Protecting Your Keys

To keep keys safe, many cryptographic algorithms use an *initialization vector (IV)*. An IV consists of random bits that the algorithm adds to the key before encrypting the data. This ensures that even data encrypted with the same key follows different encryption processes. IVs are a great way to protect against cryptanalysis, because they omit patterns from encrypted data.

Another way to protect keys is to limit how often you reuse them. Because keys are more likely to be cracked when used frequently, it's best to change them as often as possible. You can do this by using a *session key*, which encrypts data during a single session only, such as during a connection to Google. As soon as you close the Google window or a certain timeout is reached, the session is over and the platform destroys the key. That way, even if an attacker somehow obtains the key during the session, they won't be able to use it to decrypt future communications. We often use symmetric keys, such as those created during the connection to

sparklekitten.net discussed earlier in the chapter, as session keys to secure communications on the internet.

We must also periodically change asymmetric keys. Typically, this occurs on a yearly basis as part of normal certificate renewal, but you can do it more often. A good rule to adhere to when deciding when to update asymmetric keys is that the more traffic is encrypted using the key, the sooner it needs to be changed: a key used twice a year has much less exposure than one used thousands of times a month (or, in the case of a big company like Google, thousands of times a second).

How Black Hats Break Hashes

We often use hashes to check the integrity of certain data, such as a password. By their very nature, hashes aren't reversible. The hashing algorithm only creates a hash; it can't be used to decrypt it. So, most normal cryptanalysis attacks won't work on hashing algorithms. However, hashes aren't invulnerable to attack.

The main way that adversaries break hashes is by using brute-force techniques. This can be as simple as creating a table containing the hashes of random but plausible inputs and then comparing the hash they're trying to break to the hashes in the table. Once they find the hash, they know that they've correctly guessed the input. But given the amount of inputs possible—a near infinite quantity—this approach is rarely effective unless they have information to narrow down the possible results, perhaps by using other password-cracking techniques, like a dictionary attack or social engineering. But hashes do have a critical flaw that makes them easier to brute-force than other encryption keys: that flaw is *collisions*.

A collision occurs when two inputs create the same hash output. For example, by random chance, the words *sparkle* and *kitten* might both create the hash `f90ab7`. Collisions in hashing algorithms reduce the amount of time it takes to reverse a hash. Think of it this way: if you had a list of 10 items that you had to get at the grocery store, and each item was in a different part of the store, it would take you, say, 20 minutes to find them all. But if two of the items were right next to each other on the shelf, it would reduce the time it took to find all the items. The more items that were next to each other, the less time it would take to find them all. The more collisions there are in the hashing algorithm, the less time it takes an attacker to go through the various combinations to find the one they're looking for. Finding a collision can effectively halve the amount of time necessary to guess the input given a specific hash output. More collisions reduce that time further.

Stronger hashing algorithms generate fewer collisions and thus take longer to break. But as a workaround, black hats use a technique that requires a special tool known as a *rainbow table*. Essentially, a rainbow table is a table of precomputed hashes that have been grouped together by hash output to make a specific output easier to find. So if I have the hash `4fd3cd`, the table

will attempt to isolate that hash into a specific group of outputs and find the password that created the hash in that group. Once the table finds the password, I can run it through the hashing algorithm to confirm it's correct.

Salting Your Hashes

The MD5 algorithm is extremely susceptible to collision attacks. It can take seconds, given the right information, to break one of its hashes. SHA-1, although more secure than MD5, is also susceptible. SHA-2 and SHA-3 are more secure due to their hash size. Another way to strengthen hashes is to use salt. Like an IV, a salt is a series of random bits that we add to an input before hashing it. This procedure creates a unique hash that is completely different from the unsalted input.

As an example, let's say the input *kitten* creates the hash `f903d`. To keep the hash more secure, my system might hash *kitten* by adding a random set of numbers to the end, making the input something like *kitten123* (although, in practice, it would add bits to the input's binary representation rather than decimal digits). Because the system adds new numbers to *kitten* every time it hashes that input, the hash is different every time. This process also makes it harder to group hashes together in a rainbow table because the output isn't directly connected to the original plaintext that was hashed.

By adding salt, you increase the number of possible hashes to a point at which collisions become incredibly rare or difficult to discover. As a result, breaking the hashes becomes a very long process, especially if the inputs are complex (see "Type 1: Something You Know" on page 76 for more details). That said, just like encryption keys, any hash can be broken given enough time.

Exercise: Encrypting and Hashing Files

Windows and macOS systems provide several tools to help you encrypt and hash files. In this exercise, you'll use the tools embedded in Windows 10 and macOS to encrypt a file. You'll also hash a file and then encrypt it and hash it again to see the difference between the hashed outputs. After completing the exercise, you'll be able to protect your files and verify whether they've been modified. (Only the Pro version of Windows 10 allows for inherent file encryption. For those using Windows Home Edition, I suggest using the VeraCrypt Open Source software to create a place to store sensitive files so they're encrypted.)

Encrypting and Hashing a File in Windows 10

To practice using file encryption, you need to create a file to encrypt. The quickest way to do this is to open a text editor, add some text, and then save it as a *.txt* file in an easy-to-remember location; you'll be entering the file

path later. Choose a filename without spaces to make it easier to use later in the exercise. Figure 9-4 shows a super-secret file I created called *Secretfile.txt* and saved to a folder called *Secret*.

Next, you need to make sure that no one can modify your secret file without you detecting a change. One of the easiest ways to do this is to create a hash of the file. You can then compare this hash with the hashes of other versions of the same file to confirm the contents are identical. Windows 10 features built-in tools that allow you to create such a hash. To access these tools, open the Command Prompt application, just as you did in the exercise in Chapter 2.

Figure 9-4: A very secret file

In the Command Prompt window, you'll use the certutil command line tool. Normally we use certutil to find information about certificates, but it can also create file hashes. Run the tool using the following command:

```
C:\Windows\System32> certutil -hashfile C:\Users\SparkleKitten\Documents\
Secretfile.txt SHA512

SHA512 hash of C:\Users\samgr\Desktop\Secret\SuperSecret.txt:
0dd47a4aa75835dfd19b1bb6ed5f8f60cc87492dacf8284ef598229cc258244f67d430e18d7cb
770d36ed8b205af1571f42f9956bbe544a362ca191256450eb0
CertUtil: -hashfile command completed successfully.
```

This command runs certutil using the `hashfile` function. Recall from the exercise in Chapter 2 that subsequent commands beginning with a dash (-), which are called *flags*, can modify the original command. Add the path to the file so the system knows which file you want to hash. To enter the file path, start with the hard drive letter (often it's *C*), and list each folder in order until you reach the one where you stored your file; be sure to use the backslash (\) between each folder name. Then choose the hashing algorithm to use. By default this tool uses SHA-1, so here we enter `SHA2-512` to make sure we're using a strong algorithm. Press ENTER to run the command.

The command calculates the hash and then outputs a long string of characters. Copy and paste this string into another *.txt* file, such as one called *SecretHash.txt*, to save it for later comparison.

Now that you've hashed the file, you need to encrypt it. For this exercise, you'll use the built-in encryption feature in Windows 10. To access this feature, right-click the file in the filesystem, and click **Properties** to open the Properties menu. Find the area labeled Attributes at the bottom of the window and click **Advanced**. From here, you'll see a few different options. Select **Encrypt Contents To Secure Data**, as shown in Figure 9-5, and then click **OK** to encrypt the file using the default AES algorithm.

Figure 9-5: Selecting the encrypt contents option

Now the file is encrypted and secure. Let's run the certutil tool one more time to compare the hashes of the original (*Secretfile.txt*) and encrypted versions of the file. Because you already have the hash for *Secretfile.txt* when it wasn't encrypted in the *SecretHash.txt* file, all you need to do now is create a new hash for *Secretfile.txt* that is encrypted. Keep in mind that any change to the file, no matter how slight, should create a new hash. After running the same `certutil` command as you did earlier, paste the hash into the text file called *SecretHash.txt* where you pasted the first hash and compare the two. Notice that the hashes are vastly different, which indicates that something happened to the original file. If you hadn't made that change, you'd suspect the file had been tampered with for malicious purposes.

Protecting Files Using macOS

Protecting files in macOS is simple because you can access every tool you need from the Terminal using a few basic commands. To start, you'll need

to create a file to use as an example. Open a text editor, create a new file named *Secret*, and save it as a *.rtf* file in your *Documents* folder. You can write any message you want in the file.

Once you've created the file, you can hash it to give you a baseline to check against in case the file is modified. Open the Terminal application, as you did in the Chapter 2 exercise. You don't need any special permissions to use the commands in the Terminal.

To hash *Secret.rtf*, use the following `shasum` command. To indicate the home directory, you can use the tilde (~) symbol instead of typing it in the path:

```
$ shasum ~/Documents/Secret.rtf

2966acd0faf387e024b8b6be50f47450c3c2f7fb  /Users/sparklekitten/Documents/
Secret.rtf
```

Once you've entered the command, it automatically produces a hash. The long string of characters is the hash of your file. Copy this string into a new file called *SecretHash.rtf* to save it for later comparison.

To encrypt *Secret.rtf*, you'll use the openssl tool. SSL is a form of encryption for network communication, and openssl is an open source toolset for that protocol. You can use it to encrypt *Secret.rtf* by entering the following command:

```
$ openssl aes256 -in ~/Documents/Secret.rtf -out ~/Documents/Secret.rtf.enc

enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

This command uses the AES-256 algorithm, which, as you learned in this chapter, is incredibly strong. It then takes an input—in this case, the file you want to encrypt. The output uses a different filename to distinguish between the original *Secret.rtf* file and the encrypted one, so I suggest adding *.enc* to the end of the encrypted file's name so you know it's encrypted. After you press ENTER to run the command, you'll be asked to enter a passphrase twice (the second time is a confirmation). Be sure to remember this passphrase, because it's the only way to decrypt your file.

To decrypt *Secret.rtf*, you run the same command again but add `-d` for decryption and reverse the `-in` and `-out` paths, as shown here:

```
$ openssl aes256 -d -in ~/Documents/Secret.rtf.enc -out ~/Documents/Secret.rtf
```

For now, let's leave *Secret.rtf* encrypted so you can hash it again and compare it with the original hash you created before encrypting the file. Run the `shasum` command again, but this time, point it to the encrypted file:

```
$ shasum -/Documents/Secret.rtf.enc
786109556539fa6571704db78b79fb0d6ae035db
```

As you can see, the hash is completely different from the original you created and saved into your *SecretHash.rtf* file. This hash will help you detect whether your file has been modified, which might indicate that something malicious was added to it.

Using ssh-keygen to Generate a Public Key (Windows 10 or macOS)

Now that you've encrypted your file, whether on Windows 10 or macOS, and created your hashes, you'll need to create a secure means of sending the file. Even though it's encrypted, it's always best to send a file using a secure communication channel. To do this, you might need a set of public/private asymmetric keys. You can generate an RSA key pair using the command `ssh-keygen`, whether you're on Windows or macOS. For this exercise, I used the Windows command, but it's the same command for macOS:

```
C:\Windows\System32> ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\samgr/.ssh/id_rsa): mykey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in mykey.
Your public key has been saved in mykey.pub.
The key fingerprint is:
SHA256:FCRaZDnraock8vueS1FqjEZmdzcRB+LqXzRvRwqrLxc samgr@DESKTOP-OPFVANO
The key's randomart image is:
+---[RSA 2048]----+
|     .*o*o.      |
|     =oo +       |
|  + o +o+        |
| + + =.o .       |
|  o *.  S   .    |
|...o....E= o     |
| o ooo  o.+ .    |
|  ..+ooo.. .     |
|  .==oo+.        |
+----[SHA256]-----+
```

A prompt appears asking where you want to store the file. The default path is the *.ssh* folder in the current user's directory (in the example the directory is *samgr*, but it will be whatever the directory is named on your system). The default filename is *id_rsa*. To select this default press ENTER, or create your own file path if you want to save the key in a particular place. The next prompt asks for a passphrase to secure the key against misuse. If you add a passphrase, you'll need to enter that phrase every time you use the key. Enter a strong passphrase and then reenter it at the next prompt. Once you do this, your key is generated and saved to the *.ssh* folder in your *Users* directory.

The *.ssh* folder should now contain two files. One is *id_rsa*, which is the actual private key. You can open this file in Notepad to view the private key. The other file is a publisher file that contains the public key. You can also

open this file in Notepad to see the public key. Now you have a public/private key pair that you can use for encrypted communication when necessary.

Knowing how to create a hash, encrypt a file, and generate public/private key pairs allows you to encrypt your communications to ensure their safety. Using these techniques, you can protect your files from unauthorized access and determine whether they were modified. You can also check files you download, especially executables, to see whether their hash matches what the vendor posts as the real hash. If it's different, it's possible a black hat changed the file to add malware or other malicious code to it.

Conclusion

Cryptography is a complex subject with lots of moving parts. This makes it challenging to know how to keep your communications secure. By recognizing the basic structure of how cryptography is used in the modern world, with symmetric and asymmetric encryption and hashing working together to create secure connections, you'll have a better understanding of how to keep your connections secure from adversaries trying to steal your data. Although there are numerous ways that black hats can steal keys or crack encryption, using cryptography correctly works well to mitigate the risk of an attack being successful. One easy way to ensure that it's properly implemented when you're connecting to a website or using a protocol is to *check for the S*.

What do I mean by check for the S? Many protocols use the letter S to indicate that they're secure—for example, SSH (for secure remote access), FTPS (for secure file transfer), and HTTPS (for secure web connections). By contrast, Telnet, FTP, and HTTP provide the same services without encryption. Although not all secure protocols have an S in their acronym (WPA is a prime example), looking for the S is a good reminder to check for encryption when sending or storing important data. The context in which you're using the protocol determines where you need to look for the S. For example, when browsing the internet, look at the website name to ensure it starts with HTTPS instead of HTTP.

It's up to you to verify that your system sets up the encryption correctly, whether you're shopping online or sending sensitive tax documents to your accountant. You do this in the security settings, either on your computer or in a software application you're using. By understanding how encryption works, you can do a better job of ensuring it's working the way it's supposed to. This guarantees that whatever you're doing or sending is hidden from attackers, as long as it remains encrypted.

10

HOW TO DEFEAT BLACK HATS

So far in this book, you've learned what cybersecurity is, why black hats want to steal your data, how adversaries attack your systems in various ways, and what you can do to stop specific types of attacks. You should have a good idea of the threats you might face across various landscapes, such as your networks, social media, email, and more. However, as you've probably guessed by now, quality security can't be reactive. If you wait for an attack to happen before trying to prevent it, you've already lost. It's like playing a game of *Whac-A-Mole*: you might hit one mole, but unless you fix all the holes, another mole will pop up soon enough.

To have truly effective security, you need to build it from the ground up. That means starting with a plan that details how you'll create, maintain, and update your security at the outset. In this chapter, you'll learn how to create a security plan that is proactive, putting all the elements you've learned so far into action. At the end, you'll create a custom plan to protect a house, school,

business, or any other entity. Practicing how to create a security plan will help you connect everything you've learned about in this book. By the end of this chapter, you'll be ready to start developing a security strategy as a major element of any undertaking at work or at home.

What's the Worst That Could Happen?

When you're beginning any sort of security plan or even just thinking about security in general, you should always start by envisioning the worst-case scenario. Indeed, security people are often characterized as pessimists because they're constantly considering what could go wrong in any given situation, often to a degree that might seem unnecessary. But it's this type of thinking that will help you understand where your system's threats are and the risks it faces. You can only solve a problem once you identify what it involves. To start, you need to know the difference between a risk and a threat.

At first glance, risks and threats seem to have the same meaning in that they can both wreak havoc with your system or your organization. In theory, they both indicate ways your organization can be attacked or damaged. But in practice, risks and threats are distinct issues, and you must mitigate them in different ways.

Risks

A *risk* generally occurs because of something you do. You can think of a risk as the possibility that a dangerous event could happen as a result of a person or organization's actions. For example, consider getting out of bed in the morning. There is a risk that you'll twist your ankle and fall on the floor. There's a risk you'll step on a LEGO and hurt the bottom of your foot. There's even a risk that a grizzly bear will jump out of your closet and attack you as soon you get up. Now, you might think, there is no way a grizzly bear will be in my closet. You might not even live near where grizzly bears are known to roam. But that doesn't mean that it's impossible. It just means it's not very likely. In cybersecurity, there is no such thing as a risk-free action.

We calculate risk by multiplying the likelihood of an event by the impact of that event. Let's return to the getting out of bed example. What's the likelihood that you'll step on a LEGO piece when you get out of bed? If you have young children, the likelihood increases. If they own LEGO sets or have friends who do, the likelihood increases even more. If they play with LEGO bricks in your bedroom, it increases even further. But what is the impact? A hurt foot, yes, but probably nothing more than some short-term pain. If you combine the likelihood with the impact, you might be looking at a moderate risk that you will step on a LEGO brick. This risk is high enough to make you check around your bed before you go to sleep but not enough to ban LEGO within 200 yards of your house.

It helps to use numbers when calculating risk to give you an idea of how some risks compare to others. Although there are formal ways to measure risk, such as by the amount of money a risk will cost if realized, you don't have to use a standard. For example, you could choose a number between 1

and 5 to measure the likelihood and the impact. In the example of the grizzly bear in your closet, you might rate the likelihood as 1 because the possibility is extremely low. However, the impact of a grizzly bear attack would be severe, so you might rate the impact as 5. Multiplying these two ratings results in a total risk of 5.

Although a 5 rating might seem high, it can get a lot higher. Let's look at the risk of stepping on a piece of LEGO. If you have a lot of LEGO bricks in your house, the likelihood of one being near your bed is probably a 3. But the impact is a bit lower than a grizzly bear attack; let's say a 3 as well (I mean, it does really hurt to step on a LEGO piece). So the risk total is 9. That's almost double the risk of the grizzly bear attack!

Table 10-1 provides a breakdown of these calculations.

Table 10-1: Calculating Risk for Getting Out of Bed

Risk	Likelihood	Impact	Total risk
Twisted ankle	3	4	12
Step on LEGO	3	3	9
Bear attack	1	5	5

After you rate each risk, you must then determine how you'll handle that risk. When you discover a risk, you should always deal with it immediately. Take special note of a legal term called *due diligence*, which means to do what a prudent man would do. Essentially, if you know something is a risk, don't ignore it, or you could be held legally responsible should the risk be realized.

That said, there are several ways you can deal with risk:

Avoid the risk Don't do the act that causes the risk. If you don't get out of bed, you won't step on a piece of LEGO.

Transfer the risk Give ownership of the risk to another group or entity. Usually, this relates to insurance. For example, if there's a chance your house might flood, you can transfer that risk by buying flood insurance. Then, if the risk is realized, the insurance company must handle it by paying for the damage.

Mitigate the risk Do something that reduces the impact or likelihood of the risk to an acceptable level. For example, if you look around your bed for loose LEGO bricks before you go to sleep or make a rule banning them from the bedroom, you're mitigating the risk of stepping on them.

Accept the risk Accept that a bad event might happen. This is usually only done when either the impact or the likelihood of the risk is so low that it's not worth the time to try and mitigate it. For example, the likelihood of a bear attack in my bedroom is so low, there's no reason to take precautions against it.

Let's consider these options in a cybersecurity context. For instance, think about the risk that one of your employees might click a link in a

phishing email. This is a definite risk, given that nearly every business uses email in some manner. As you learned in Chapter 3, you can also regard this risk as having a high impact, especially if the victim downloads malware or provides the attacker with their credentials.

It's probably unreasonable to *avoid* this risk, because eliminating email entirely isn't a viable option for most businesses. You could *transfer* the risk somewhat by buying cybersecurity insurance, but this likely won't cover the entire impact of the risk if realized. You can't *accept* the risk either, because the impact of a realized event could be so devastating, it might shut down your business. In this case, the best option is to *mitigate* the risk by setting up spam filters and training your employees to recognize phishing attempts.

Often, managing risk isn't this cut and dried; many factors go into deciding the likelihood, impact, and strategy for dealing with risks. However, even thinking about risk in simple terms, as they're described here, can help you better understand the problems you might encounter.

Threats

A *threat* is a negative impact on a system, person, or organization. In other words, a threat is the impetus that causes bad events to happen. In the getting out of bed example, the threats are the factors that might cause you harm. In the LEGO and bear attack situations, the threat is obvious. But in the twisted ankle scenario, identifying the threat can be tricky, because it depends on what might cause you to twist your ankle. If it's a misplaced shoe, that shoe is the threat. If it's just your own clumsiness, well, then you're the threat (or, at least, the part of your brain controlling your movement is).

Threats come in all shapes and sizes. They don't necessarily have to be malicious or even conscious. For example, buildings are constantly faced with the threat that they might catch fire and burn down. But human or not, we call the agent that brings about the threat a *threat actor*. The distinction between threat and threat actor might seem silly to make, but it becomes important when you're involved in threat management; managing a threat is much easier than managing a threat actor.

For example, say you're managing the threat that your employees will click a phishing link. You might think that the threat actor is the attacker who sent the email, but that is only partially true. The employees are potential threat actors, too, because they're the ones clicking the link to activate it. Because you can't eliminate the threat actor (what's a business without employees?), you need to reduce the threat through training and spam filters.

To better handle threats, it helps to categorize them. Doing so can give you a better idea of how to eliminate them from your environment. Table 10-2 shows a popular cybersecurity threat classification scheme known as STRIDE, which Microsoft created. The mnemonic represents six types of security threats: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privileges. Essentially, all cybersecurity attacks can be grouped into one of these threat categories. In fact, you'll probably recognize most of these threats as various attacks from the earlier chapters in this book.

Table 10-2: STRIDE Model with Targets

Threat	Target
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information disclosure	Confidentiality
Denial of service	Availability
Elevation of privileges	Authorization

By classifying attacks into categories, you can get a general idea of how they work or at least what their goals might be. For example, if I told you about a new attack called Sparkle Kitten Bite, you'd have no idea what to do about it. But if I told you about a new elevation of privileges threat called Sparkle Kitten Bite, you'd at least understand that the attack was attempting to access accounts it wasn't supposed to, most likely to execute commands in a privileged environment.

Controls

Categorizing threats and threat actors also helps you choose the best *control*, which attempts to either prevent or mitigate a threat. By now, you should have a good idea of the different types of controls used to stop various attacks. For example, you know that a DoS attack can be prevented by using redundant systems or by filtering the traffic before it gets to its target (as discussed in Chapter 6). Similarly, you know the best way to defeat a brute-force attack is to make a password so complex it would take too long for even a computer to guess it (as discussed in Chapters 5 and 9).

As with categories of threats, there are categories of controls that can help you decide which best fits your situation. For risk management, we generally categorize a control by how it attempts to protect a target from a threat. The five control categories are Administrative, Preventative, Detective, Compensating, and Corrective. Table 10-3 lists these categories as well as their purposes and examples of each.

Table 10-3: Control Categories

Category	Purpose	Example
Administrative	Provides guidance on how to conduct activities	Security awareness training, policy, procedure
Preventative	Attempts to stop unwanted activity before it happens	Firewall
Detective	Attempts to discover unwanted activity after it has happened or while it's happening	IDS, logging

(continued)

Table 10-3: Control Categories *(continued)*

Category	Purpose	Example
Compensating	Adds additional security to make up for a weakness in another control	Encryption
Corrective	Fixes another control or flaw after it's discovered	Patch management, vulnerability management

Let's consider how to deploy these controls in a real-world scenario. *Preventative* controls are self-explanatory. For example, a firewall prevents unwanted activity by blocking the connections its rules are set to deny. On the other hand, *Detective* controls usually provide a means to discover malicious activity after it has occurred, such as when you review login logs to determine whether an adversary broke into your server.

Compensating controls bulk up another control to make it more secure. Encryption is a great example of a Compensating control: if another control fails—for example, an authentication control used to protect a database—an attacker still won't be able to read the data. *Corrective* controls fix a flaw found in another control. For instance, patching a system after discovering a critical flaw is a Corrective control.

Administrative controls provide guidance to an organization on how to implement security. A good example of an Administrative control is a procedure for setting up computers for new employees. If you document the steps involved, you'll ensure that it's done the same way every time and that the correct security configurations are always put in place. Administrative controls often dictate how other controls should be set up and maintained.

Understanding risk, threats, and controls helps you handle the problems you or your organization might confront, especially those that adversaries create. But knowledge of these aspects is only one part of the process. You must also make sure you're doing your due diligence, which involves dealing with all threats and maintaining your controls. This is where a risk management program can be a major benefit.

Risk Management Programs

Managing risk can be complicated, because it requires juggling several aspects of cybersecurity at once. Not only do you have to address the threats, but you also have to maintain the control you've implemented. In addition, as a cybersecurity professional, you'll be called on to answer questions about your organization's security. You'll have to explain to people with various levels of technical aptitude why certain risks need to be addressed and how the control you've chosen will address them. This requires having a full grasp of your organization's security at all times. It's no small task indeed.

A good risk management program can be a lifesaver. The purpose of a risk management program is to track your company's risk, the threats related to it, and the controls you're using to address it. Ideally, a risk management program will be flexible enough that you can continuously update

it as changes happen within your environment. You should be able to add and remove controls as new risks emerge and resolved risks are no longer a concern. The program should also provide a template to use when you need to address the risk associated with one-time projects, such as major equipment upgrades or the opening of a new building.

To manage all of these activities, you need to use a special tool called a *risk register*, which is a system for documenting all the risks you're currently tracking. Think of it as an assignment organizer, like you might have used in school. You can use it to track the risk, how you're dealing with it, and the controls you're using to address it. The program can also be an efficient way to track risks that you haven't yet dealt with or controls that aren't fully implemented.

A risk register can be a complex piece of software that provides detailed status information, but you can also use a simple spreadsheet as a risk register. In fact, it's better to use a spreadsheet than nothing at all. Keep in mind that the point of risk management is to track how you're managing risk, which, at the very least, is necessary to meet the due diligence requirements often mandated by law. Figure 10-1 shows an example of a spreadsheet created to track risk. Let's discuss each section in more detail.

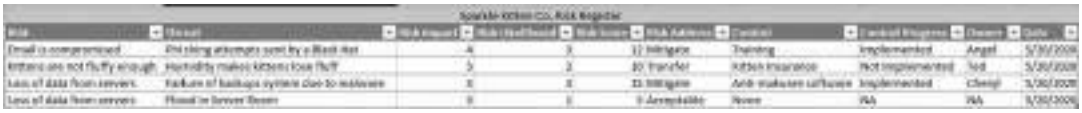


Figure 10-1: Risk register for Sparkle Kitten Inc.

The first two columns track the organization's risks and threats. Remember that risk occurs based on what we do, whereas threats are events that happen to us. In this example, because Sparkle Kitten Inc. uses email, it risks its email system being compromised. In particular, the concern is the threat of phishing attacks sent by black hats, as noted in the Threat column.

Risk registers don't always track risks *and* threats, but doing so can help distinguish between the ways a risk can be realized. For example, loss of data from your servers is always a risk when you're storing information. But notice that two different threats have been listed for this in the risk register, and they have two different outcomes based on their likelihood and impact.

The risk scores can help us determine which risks we need to address immediately and which we can move to the back burner. This risk register shows that loss of data from servers because of malware has the highest risk score. Therefore, we should address it first, because if that risk is realized, it's likely to be the most detrimental. The next two columns detail how we're addressing that risk and which control we used to address it. In the case of loss of data from servers due to malware, we mitigate the risk by using the control of anti-malware software, which we hope will detect and prevent the malware from destroying our backups.

The next three columns list the state of the control that we chose to address the risk. For example, you can see that the anti-malware software

was implemented on 5/20/2020. The *control owner*, Cheryl, is in charge of maintaining and checking that control to make sure it still adequately meets our needs. These columns also indicate what we still have to do. Notice that, although we decided to transfer the risk of our kittens losing their fluffiness by purchasing kitten insurance, we've not yet implemented this control. It will be up to Ted to buy kitten insurance for the organization. Once he does so, he'll update the register and include a new date to reflect when the insurance was put in place.

Even though this spreadsheet might be simple, it can help organize your company's risks and threats, making them easier to address. At a glance, you can see the risks you've identified, the impact of the risks, and the state of the controls process. You can also identify who you need to talk to in order to get a status on a control. The best part is that you don't need detailed technical knowledge to understand that your organization is safe.

Putting It All Together

Let's combine everything you've learned in this book into a single example. Say you work as a security analyst for a medium-sized organization with 500 employees. Part of your daily job is to check for any alerts that come in either from employees or from your firewall and intrusion detection system.

One morning, you receive an email from a panicked employee saying they received a weird email and they clicked the link inside. Now they're afraid it might have caused some issues on their computer. You check the email using some quick phishing analysis tools, such as VirusTotal and MX Toolbox, which you practiced using in Chapter 3. The email claims to be a password reset notice from Microsoft, but it comes from the email address *M1cos0ft.com*. Looking at the link, you realize that it's possibly malicious. The employee said when they clicked the link, it asked them to download a password update tool that then ran on their computer. You realize this is likely malware and begin a virus scan on the system.

While you're scanning the system, you check your firewall and IDS alerts to see whether they contain anything suspicious. Sure enough, you notice alerts about new, possibly malicious traffic coming from the employee's computer. You aren't sure what to do with this information, so you escalate the alerts to a senior security consultant on the team. They look at the alerts and realize it likely means the employee downloaded a trojan that is attempting to spread ransomware across the network. The senior security consultant quickly updates the IDS and firewall to block any attempts from the employee's computer to make connections to other computers on the network. Meanwhile, your scan picks up a well-known malware kit. You're able to isolate it and delete the infection, but you completely reset the employee's computer just to be sure.

After the incident, you and your fellow security co-workers do a post-incident analysis to discuss what happened and how to prevent it in the future. The CISO leads this meeting. From the discussion, everyone realizes that the organization has a weak point: training employees on how to recognize phishing emails, which are a major threat, as evidenced by the ransomware attack. The CISO adds this to the company's risk register

and discusses how best to implement the control of training employees. Everyone agrees that a special phishing training platform is the best way to implement this control.

The CISO brings the new risk to the quarterly risk management meeting and discusses the idea of purchasing a phishing training platform for the organization. The head of HR agrees with the CISO's idea, because the current training platform that HR uses doesn't include information about phishing. The CFO also agrees because the cost is low but the benefit would be immense. It's decided that the CISO will investigate various options and report back to the committee in a month with a recommendation. The CISO updates the risk register and assigns a security team member to help with this task.

Although this is a fictional scenario, it represents how security should function in an organization. In the scenario, it wasn't enough to manage an employee's call about a phishing email. The security analyst had to cross-reference different sources of information and request advice from other team members to see the whole picture. Also, the work didn't stop once the incident was over. Just as important was the follow-up, which allowed the organization to recognize its failings in security and fix them. Using the expertise of the security team, the CISO successfully showed why it was important to work with senior leadership to purchase the special training platform. This is why a risk management plan is critical: it helps tie every aspect of security together. It shows the need for security and the solution in a simple, easy-to-manage format.

Exercise: Conducting a Risk Analysis

For this final exercise, choose a target and conduct a risk analysis against it. A risk analysis defines all the risks to a target and examines the risk management process for each of these. Your target can be your home, your school, your place of work, or any other location where you have a good idea of the cybersecurity threats or risks that exist. Once you've chosen your target, complete the following steps:

1. Identify all the assets you want to include in your analysis. For example, in your home, you might list any computers, network devices like routers, and smart devices like game consoles or TVs. List these devices in a spreadsheet or on a piece of paper.
2. Record all the ways you think those assets could be attacked or otherwise harmed. When doing this, remember to perform the sanity check of considering the likelihood of an attack. Yes, in the movies a rogue AI might take over your game console and try to kill you, but in real life, that's probably not something to worry about.
3. Look at all the ways your assets can be attacked and group them using the STRIDE model. Identify what the attacks have in common. These are the threats to your target. For example, your TV and game console might be susceptible to DoS attacks.

4. Determine which of your groupings have the highest number of instances. The higher the number of instances, the higher the likelihood. Also, add a quick note on the potential impact. For example, if your game console is hit with a DoS attack, you won't be able to play the new video game that's just come out with your friends, which will be extremely disappointing, giving it a high impact.
5. Place the threats into a risk register, such as the one shown earlier in this chapter. Include what the risk is, the threat, and the risk score.
6. Look at which controls you have in place to deal with these attacks. For example, in the case of a DoS attack against your game console, you might research how much bandwidth you would need to handle that sort of attack, what sort of security your console has, or what protections your ISP has in place.
7. Complete the risk register with how you're dealing with the risk and which controls you're using to address the risk.

Now you have a completed risk register that gives you a good idea of the threats your target faces and what you can do to address them, if you haven't already. Although it might not always be practical to mitigate the threats identified (it's very hard to stop a DoS attack on your own), the risk register provides you with a way to practice your risk management scenario.

Farewell and Good Luck

Now you're ready to start your journey into the world of cybersecurity. Whether you intend to join the ranks of cybersecurity professionals or just want to apply this new knowledge to your everyday life, you've acquired a strong foundation that you can use to explore the security topics that interest you.

Here are some final tips for your security journey:

- Think twice before you click. Even the best professionals get fooled when they're in a rush. Cybersecurity is slow but steady.
- Make the time to take the proper steps. Cybersecurity work might sometimes seem like a task that should have been dealt with yesterday; nevertheless, take a breather to choose the best next step.
- Never take someone's word for it if the situation doesn't feel right to you. If you're not sure a configuration is done right, don't assume another person caught it.
- Always ask for help. Cybersecurity isn't a vacuum, nor is it limited to your team.
- Keep reading and learning. Cybersecurity requires constant upkeep to remain one step ahead of black hats.
- Have fun! Cybersecurity is serious, but that doesn't mean it has to be *serious*.

INDEX

Italicized page numbers indicate definitions of terms.

C

Never before has the world relied so heavily on the Internet to stay connected and informed. That makes the Electronic Frontier Foundation's mission—to ensure that technology supports freedom, justice, and innovation for all people—more urgent than ever.

For over 30 years, EFF has fought for tech users through activism, in the courts, and by developing software to overcome obstacles to your privacy, security, and free expression. This dedication empowers all of us through darkness. With your help we can navigate toward a brighter digital future.

ELECTRONIC FRONTIER FOUNDATION EFF

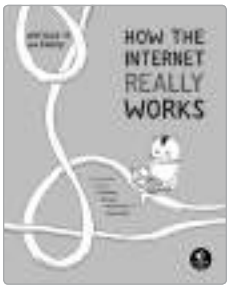
LEARN MORE AND JOIN EFF AT EFF.ORG/NO-STARCH-PRESS

RESOURCES

Visit *https://nostarch.com/cybersecurityreallyworks/* for errata and more information.

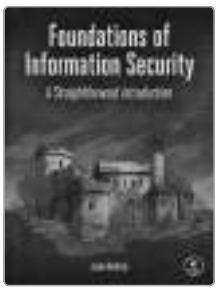
More no-nonsense books from **NO STARCH PRESS**

HOW COMPUTERS REALLY WORK
A Hands-On Guide to the Inner Workings of the Machine
BY MATTHEW JUSTICE
380 PP., $39.95
ISBN 978-1-7185-0066-2

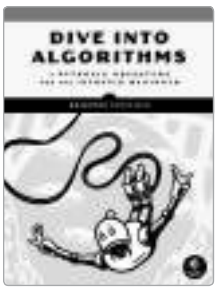


HOW THE INTERNET REALLY WORKS
An Illustrated Guide to Protocols, Privacy, Censorship, and Governance
BY ARTICLE 19
120 PP., $19.95
ISBN 978-1-7185-0029-7

BLACK HAT PYTHON, 2ND EDITION
Python Programming for Hackers and Pentesters
BY JUSTIN SEITZ AND TIM ARNOLD
216 PP., $44.99
ISBN 978-1-7185-0112-6



FOUNDATIONS OF INFORMATION SECURITY
A Straightforward Introduction
BY JASON ANDRESS
248 PP., $39.95
ISBN 978-1-7185-0004-4



DIVE INTO ALGORITHMS
A Pythonic Adventure for the Intrepid Beginner
BY BRADFORD TUCKFIELD
248 PP., $39.95
ISBN 978-1-7185-0068-6

LINUX BASICS FOR HACKERS
Getting Started with Networking, Scripting, and Security in Kali
BY OCCUPYTHEWEB
248 PP., $34.95
ISBN 978-1-59327-855-7

PHONE:
800.420.7240 OR
415.863.9900

EMAIL:
SALES@NOSTARCH.COM
WEB:
WWW.NOSTARCH.COM

PLEASE DON'T FEED THE TROLLS

How Cybersecurity Really Works is the perfect introduction to cybersecurity. Whether you're a computer science student or a business professional, it will teach you the basics without all the jargon.

This beginners guide covers different types of attacks, common tactics used by online adversaries, and defensive strategies you can use to protect yourself. You'll learn what security professionals do, what an attack looks like from a cybercriminal's viewpoint, and how to implement sophisticated cybersecurity measures on your own devices.

In addition, you'll find explanations of topics like malware, phishing, and social engineering attacks, coupled with real-world examples and hands-on exercises to help you apply what you've learned. You'll explore ways to bypass access controls, prevent infections from worms and viruses, and protect your cloud accounts from attackers.

You'll also learn how to:

- Analyze emails to detect phishing attempts
- Use SQL injection to attack a website
- Examine malware from the safety of a sandbox environment
- Use the command line to evaluate and improve your computer and network security
- Deploy encryption and hashing to protect your files
- Create a comprehensive risk management plan

You can't afford to ignore cybersecurity anymore, but attackers won't wait while you read a long technical manual. That's why *How Cybersecurity Really Works* teaches you just the essentials you need to think beyond antivirus and make the right decisions to keep the online monsters at bay.

ABOUT THE AUTHOR

Sam Grubb is a cybersecurity consultant and education advocate who works with companies and healthcare providers to ensure they are meeting both security and compliance needs. He has over six years of experience teaching cybersecurity and holds several cybersecurity certifications. He lives in Arkansas with his wife, son, two cats, and two dogs.

no starch press

THE FINEST IN GEEK ENTERTAINMENT™

www.nostarch.com

$24.99 ($33.99 CDN)

ISBN 978-1-7185-0128-7

9 781718 501287 52499