CHAPTER 16

Security

Both protection and security are vital to computer systems. We distinguish between these two concepts in the following way: Security is a measure of confidence that the integrity of a system and its data will be preserved. Protection is the set of mechanisms that control the access of processes and users to the resources defined by a computer system. We focus on security in this chapter and address protection in Chapter 17.

Security involves guarding computer resources against unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency. Computer resources include the information stored in the system (both data and code), as well as the CPU, memory, secondary storage, tertiary storage, and networking that compose the computer facility. In this chapter, we start by examining ways in which resources may be accidentally or purposely misused. We then explore a key security enabler—cryptography. Finally, we look at mechanisms to guard against or detect attacks.

CHAPTER OBJECTIVES

- Discuss security threats and attacks.
- Explain the fundamentals of encryption, authentication, and hashing.
- Examine the uses of cryptography in computing.
- Describe various countermeasures to security attacks.

16.1 The Security Problem

In many applications, ensuring the security of the computer system is worth considerable effort. Large commercial systems containing payroll or other financial data are inviting targets to thieves. Systems that contain data pertaining to corporate operations may be of interest to unscrupulous competitors. Furthermore, loss of such data, whether by accident or fraud, can seriously impair the ability of the corporation to function. Even raw computing resources are attractive to attackers for bitcoin mining, for sending spam, and as a source from which to anonymously attack other systems.

In Chapter 17, we discuss mechanisms that the operating system can provide (with appropriate aid from the hardware) that allow users to protect their resources, including programs and data. These mechanisms work well only as long as the users conform to the intended use of and access to these resources.

We say that a system is **secure** if its resources are used and accessed as intended under all circumstances. Unfortunately, total security cannot be achieved. Nonetheless, we must have mechanisms to make security breaches a rare occurrence, rather than the norm.

Security violations (or misuse) of the system can be categorized as intentional (malicious) or accidental. It is easier to protect against accidental misuse than against malicious misuse. For the most part, protection mechanisms are the core of accident avoidance. The following list includes several forms of accidental and malicious security violations. Note that in our discussion of security, we use the terms **intruder**, **hacker**, and **attacker** for those attempting to breach security. In addition, a **threat** is the potential for a security violation, such as the discovery of a vulnerability, whereas an **attack** is an attempt to break security.

- **Breach of confidentialit** . This type of violation involves unauthorized reading of data (or theft of information). Typically, a breach of confidentiality is the goal of an intruder. Capturing secret data from a system or a data stream, such as credit-card information or identity information for identity theft, or unreleased movies or scripts, can result directly in money for the intruder and embarrassment for the hacked institution.
- **Breach of integrity**. This violation involves unauthorized modification of data. Such attacks can, for example, result in passing of liability to an innocent party or modification of the source code of an important commercial or open-source application.
- **Breach of availability**. This violation involves unauthorized destruction of data. Some attackers would rather wreak havoc and get status or bragging rights than gain financially. Website defacement is a common example of this type of security breach.
- **Theft of service**. This violation involves unauthorized use of resources. For example, an intruder (or intrusion program) may install a daemon on a system that acts as a file server.
- **Denial of service**. This violation involves preventing legitimate use of the system. **Denial-of-service** (**DOS**) attacks are sometimes accidental. The original Internet worm turned into a DOS attack when a bug failed to delay its rapid spread. We discuss DOS attacks further in Section 16.3.2.

Attackers use several standard methods in their attempts to breach security. The most common is **masquerading**, in which one participant in a communication pretends to be someone else (another host or another person). By masquerading, attackers breach **authentication**, the correctness of identification; they can then gain access that they would not normally be allowed. Another common attack is to replay a captured exchange of data. A **replay attack** consists of the malicious or fraudulent repeat of a valid data transmission. Sometimes the replay comprises the entire attack—for example, in a repeat of a request to transfer money. But frequently it is done along with **message**

modificatio , in which the attacker changes data in a communication without the sender's knowledge. Consider the damage that could be done if a request for authentication had a legitimate user's information replaced with an unauthorized user's. Yet another kind of attack is the **man-in-the-middle attack**, in which an attacker sits in the data flow of a communication, masquerading as the sender to the receiver, and vice versa. In a network communication, a man-in-the-middle attack may be preceded by a **session hijacking**, in which an active communication session is intercepted.

Another broad class of attacks is aimed at **privilege escalation**. Every system assigns privileges to users, even if there is just one user and that user is the administrator. Generally, the system includes several sets of privileges, one for each user account and some for the system. Frequently, privileges are also assigned to nonusers of the system (such as users from across the Internet accessing a web page without logging in or anonymous users of services such as file transfer). Even a sender of email to a remote system can be considered to have privileges—the privilege of sending an email to a receiving user on that system. Privilege escalation gives attackers more privileges than they are supposed to have. For example, an email containing a script or macro that is executed exceeds the email sender's privileges. Masquerading and message modification, mentioned above, are often done to escalate privileges. There are many more examples, as this is a very common type of attack. Indeed, it is difficult to detect and prevent all of the various attacks in this category.

As we have already suggested, absolute protection of the system from malicious abuse is not possible, but the cost to the perpetrator can be made sufficiently high to deter most intruders. In some cases, such as a denial-of-service attack, it is preferable to prevent the attack but sufficient to detect it so that countermeasures can be taken (such as up-stream filtering or adding resources such that the attack is not denying services to legitimate users).

To protect a system, we must take security measures at four levels:

1. **Physical**. The site or sites containing the computer systems must be physically secured against entry by intruders. Both the machine rooms and the terminals or computers that have access to the target machines must be secured, for example by limiting access to the building they reside in, or locking them to the desk on which they sit.

2. **Network**. Most contemporary computer systems—from servers to mobile devices to Internet of Things (IoT) devices—are networked. Networking provides a means for the system to access external resources but also provides a potential vector for unauthorized access to the system itself.

   Further, computer data in modern systems frequently travel over private leased lines, shared lines like the Internet, wireless connections, and dial-up lines. Intercepting these data can be just as harmful as breaking into a computer, and interruption of communications can constitute a remote denial-of-service attack, diminishing users' use of and trust in the system.

3. **Operating system**. The operating system and its built-in set of applications and services comprise a huge code base that may harbor many vulnerabilities. Insecure default settings, misconfigurations, and security

bugs are only a few potential problems. Operating systems must thus be kept up to date (via continuous patching) and "hardened"—configured and modified to decrease the attack surface and avoid penetration. The **attack surface** is the set of points at which an attacker can try to break into the system.

4. **Application**. Third-party applications may also pose risks, especially if they possess significant privileges. Some applications are inherently malicious, but even benign applications may contain security bugs. Due to the vast number of third-party applications and their disparate code bases, it is virtually impossible to ensure that all such applications are secure.

This four-layered security model is shown in Figure 16.1.

The four-layer model of security is like a chain made of links: a vulnerability in any of its layers can lead to full system compromise. In that respect, the old adage that security is only as strong as its weakest link holds true.

Another factor that cannot be overlooked is the human one. Authorization must be performed carefully to ensure that only allowed, trusted users have access to the system. Even authorized users, however, may be malicious or may be "encouraged" to let others use their access—whether willingly or when duped through **social engineering**, which uses deception to persuade people to give up confidential information. One type of social-engineering attack is **phishing**, in which a legitimate-looking e-mail or web page misleads a user into entering confidential information. Sometimes, all it takes is a click of a link on a browser page or in an email to inadvertently download a malicious payload, compromising system security on the user's computer. Usually that PC is not the end target, but rather some more valuable resource. From that compromised system, attacks on other systems on the LAN or other users ensue.

So far, we've seen that all four factors in the four-level model, plus the human factor, must be taken into account if security is to be maintained. Furthermore, the system must provide protection (discussed in great detail in Chapter 17) to allow the implementation of security features. Without the ability to authorize users and processes to control their access, and to log their activities, it would be impossible for an operating system to implement security measures or to run securely. Hardware protection features are needed to support an overall protection scheme. For example, a system without memory

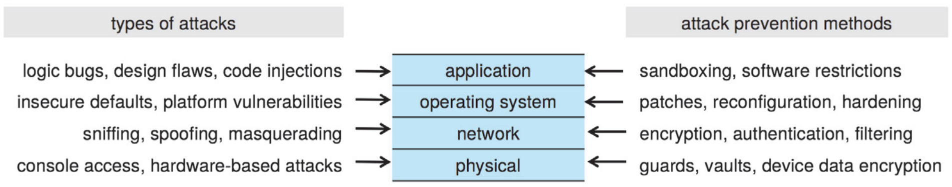
types of attacks
attack prevention methods
logic bugs, design flaws, code injections
application
sandboxing, software restrictions
insecure defaults, platform vulnerabilities
operating system
patches, reconfiguration, hardening
sniffing, spoofing, masquerading
network
encryption, authentication, filtering
console access, hardware-based attacks
physical
guards, vaults, device data encryption

Figure 16.1 The four-layered model of security.

protection cannot be secure. New hardware features are allowing systems to be made more secure, as we shall discuss.

Unfortunately, little in security is straightforward. As intruders exploit security vulnerabilities, security countermeasures are created and deployed. This causes intruders to become more sophisticated in their attacks. For example, spyware can provide a conduit for spam through innocent systems (we discuss this practice in Section 16.2), which in turn can deliver phishing attacks to other targets. This cat-and-mouse game is likely to continue, with more security tools needed to block the escalating intruder techniques and activities.

In the remainder of this chapter, we address security at the network and operating-system levels. Security at the application, physical and human levels, although important, is for the most part beyond the scope of this text. Security within the operating system and between operating systems is implemented in several ways, ranging from passwords for authentication through guarding against viruses to detecting intrusions. We start with an exploration of security threats.

16.2 Program Threats

Processes, along with the kernel, are the only means of accomplishing work on a computer. Therefore, writing a program that creates a breach of security, or causing a normal process to change its behavior and create a breach, is a common goal of attackers. In fact, even most nonprogram security events have as their goal causing a program threat. For example, while it is useful to log in to a system without authorization, it is quite a lot more useful to leave behind a back-door daemon or **Remote Access Tool** (**RAT**) that provides information or allows easy access even if the original exploit is blocked. In this section, we describe common methods by which programs cause security breaches. Note that there is considerable variation in the naming conventions for security holes and that we use the most common or descriptive terms.

16.2.1 Malware

Malware is software designed to exploit, disable or damage computer systems. There are many ways to perform such activities, and we explore the major variations in this section.

Many systems have mechanisms for allowing programs written by a user to be executed by other users. If these programs are executed in a domain that provides the access rights of the executing user, the other users may misuse these rights. A program that acts in a clandestine or malicious manner, rather than simply performing its stated function, is called a **Trojan horse**. If the program is executed in another domain, it can escalate privileges. As an example, consider a mobile app that purports to provide some benign functionality—say, a flashlight app—but that meanwhile surreptitiously accesses the user's contacts or messages and smuggles them to some remote server.

A classic variation of the Trojan horse is a "Trojan mule" program that emulates a login program. An unsuspecting user starts to log in at a terminal, computer, or web page and notices that she has apparently mistyped her