testing and analysis Patch generation 5. Possible x generation 3. Forward features 6. Application update 4. Vulnerability testing and identi cation 1. Worm scans or infection attempts 2. Noti cations 368 chapter 10 / Malicious Software DDoS Attack Description A DDoS attack attempts to consume the target's resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked. A simple example of an internal resource attack is the SYN flood attack. Figure 10.5a shows the steps involved: 1. The attacker takes control of multiple hosts over the Internet, instructing them to contact the target Web server. 2. The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target. 3. Each SYN packet is a request to open a TCP connection. For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowledge) packet, trying to establish a TCP connection with a TCP entity at a spurious IP address. The Web server maintains a data structure for each SYN request waiting for a response back and becomes bogged down as more traffic floods in. The result is that legitimate connections are denied while the victim machine is waiting to complete bogus "half-open" connections. The TCP state data structure is a popular internal resource target but by no means the only one. [CERT01] gives the following examples: 1. An intruder may attempt to use up available data structures that are used by the OS to manage processes, such as process table entries and process control information entries. The attack can be quite simple, such as a program that forks new processes repeatedly. 2. An intruder may attempt to allocate to itself large amounts of disk space by a variety of straightforward means. These include generating numerous e-mails, forcing errors that trigger audit trails, and placing files in shareable areas. Figure 10.5b illustrates an example of an attack that consumes data transmission resources. The following steps are involved: 1. The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP ECHO packets5 with the target's spoofed IP address to a group of hosts that act as reflectors, as described subsequently. 2. Nodes at the bounce site receive multiple spoofed requests and respond by sending echo reply packets to the target site. 3. The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic. 5 The Internet Control Message Protocol (ICMP) is an IP-level protocol for the exchange of control packets between a router and a host or between hosts. The ECHO packet requires the recipient to respond with an echo reply to check that communication is possible between entities. 10.11 / Distributed Denial of Service Attacks 369 Another way to classify DDoS attacks is as either direct or reflector DDoS attacks. In a direct DDoS attack (Figure 10.6a), the attacker is able to implant zombie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involves two levels of zombie machines: master zombies and slave zombies. The hosts of both machines have been infected with malicious code. The attacker coordinates and triggers the master zombies, which in turn coordinate and trigger the slave zombies. The use of two levels of zombies makes it more difficult to trace the attack back to its source and provides for a more resilient network of attackers. A reflector DDoS attack adds another layer of machines (Figure 10.6b). In this type of attack, the slave zombies construct packets requiring a response that contain the target's IP address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack

and hence be more damaging. Further, tracing back the attack or filtering out the attack packets is more difficult because the attack comes from widely dispersed uninfected machines. Figure 10.5 Examples of Simple DDoS Attacks SYN packets Attack machine Attack machine Re ector machines Slave servers 1 1 2 2 3 3 (a) Distributed SYN ood attack (b) Distributed ICMP attack Internet Target Web server Target router SYN packets SYN/ACK packets 370 chapter 10 / Malicious Software Constructing the Attack Network The first step in a DDoS attack is for the attacker to infect a number of machines with zombie software that will ultimately be used to carry out the attack. The essential ingredients in this phase of the attack are the following: 1. Software that can carry out the DDoS attack. The software must be able to run on a large number of machines, must be able to conceal its existence, must Figure 10.6 Types of Flooding-Based DDoS Attacks (a) Direct DDoS Attack Attacker Attacker Re ectors Victim Victim Master zombies Master zombies Slave zombies Slave zombies (b) Re ector DDoS Attack 10.11 / Distributed Denial of Service Attacks 371 be able to communicate with the attacker or have some sort of time-triggered mechanism, and must be able to launch the intended attack toward the target. 2. A vulnerability in a large number of systems. The attacker must become aware of a vulnerability that many system administrators and individual users have failed to patch and that enables the attacker to install the zombie software. 3. A strategy for locating vulnerable machines, a process known as scanning. In the scanning process, the attacker first seeks out a number of vulnerable machines and infects them. Then, typically, the zombie software that is installed in the infected machines repeats the same scanning process, until a large distributed network of infected machines is created. [MIRK04] lists the following types of scanning strategies: ■■ Random: Each compromised host probes random addresses in the IP address space, using a different seed. This technique produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched. ■■ Hit list: The attacker first compiles a long list of potential vulnerable machines. This can be a slow process done over a long period to avoid detection that an attack is underway. Once the list is compiled, the attacker begins infecting machines on the list. Each infected machine is provided with a portion of the list to scan. This strategy results in a very short scanning period, which may make it difficult to detect that infection is taking place. ■■ Topological: This method uses information contained on an infected victim machine to find more hosts to scan. ■■ Local subnet: If a host is infected behind a firewall, that host then looks for targets in its own local network. The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall. DDoS Countermeasures In general, there are three lines of defense against DDoS attacks [CHAN02]: ■■ Attack prevention and preemption (before the attack): These mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. Techniques include enforcing policies for resource consumption and providing backup resources available on demand. In addition, prevention mechanisms modify systems and protocols on the Internet to reduce the possibility of DDoS attacks. ■■ Attack detection and filtering (during the attack): These mechanisms attempt to detect the attack as it begins and respond immediately. This minimizes the impact of the attack on the target. Detection involves looking for suspicious patterns of behavior. Response involves filtering out packets likely to be part of the attack. ■■ Attack source traceback and identification (during and after the attack): This is an attempt to identify the source of the attack as a first step in preventing future attacks. However, this method typically does not yield results fast enough, if at all, to mitigate an ongoing attack. 372 chapter 10 / Malicious Software The challenge in coping with DDoS

attacks is the sheer number of ways in which they can operate. Thus, DDoS countermeasures must evolve with the threat. 10.12 Key Terms, Review Questions, and Problems Key Terms adware attack kit backdoor behavior-blocking software blended attack boot sector infector bot botnet crimeware direct DDoS attack distributed denial of service (DDoS) downloader drive-by-download e-mail virus flooders keyloggers logic bomb macro virus malicious software malware metamorphic virus mobile code parasitic virus phishing polymorphic virus ransomware reflector DDoS attack rootkit scanning spear-phishing spyware stealth virus trapdoor Trojan horse virus worm zombie zero-day exploit Review Questions 10.1 What are three broad mechanisms that malware can use to propagate? 10.2 What is a blended attack? 10.3 What are typical phases of operation of a virus or worm? 10.4 Classify viruses based on the targets they try to infect. 10.5 List the features of macro viruses that enable them to infect scripting codes. 10.6 What functions does a worm perform during the propagation phase? 10.7 Give some examples of client side vulnerabilities that can be exploited by malware? 10.8 What is an "infection vector"? 10.9 Explain the difference between a keylogger and spyware with an example. 10.10 What kind of activities can be performed by an attacker using a rootkit? What makes it difficult to detect a rootkit? 10.11 Describe some malware countermeasure elements. 10.12 List three places malware mitigation mechanisms may be located. 10.13 Briefly describe the four generations of antivirus software. 10.14 List the activities that can be monitored by "behavior-blocking software". 10.15 What is the difference between a reflector DDoS attack and a direct DDoS attack? Problems 10.1 There is a flaw in the virus program of Figure 10.1a. What is it? 10.2 The question arises as to whether it is possible to develop a program that can analyze a piece of software to determine if it is a virus. Consider that we have a program D 10.12 / Key Terms, Review Questions, and Problems 373 that is supposed to be able to do that. That is, for any program P, if we run D(P), the result returned is TRUE (P is a virus) or FALSE (P is not a virus). Now consider the following program: Program CV := { . . . main-program := {if D(CV) then goto next: else infect-executable; } next: } In the preceding program, infect-executable is a module that scans memory for executable programs and replicates itself in those programs. Determine if D can correctly decide whether CV is a virus. 10.3 The following code fragments show a sequence of virus instructions and a metamorphic version of the virus. Describe the effect produced by the metamorphic code. Original Code Metamorphic Code mov eax, 5 mov eax, 5 add eax, ebx push edx call [ebx] jmp 0x89AB swap eax, ebx call [ebx] nop 10.4 The list of passwords used by the Morris worm is provided at this book's Premium Content Web site. a. The assumption has been expressed by many people that this list represents words commonly used as passwords. Does this seem likely? Justify your answer. b. If the list does not reflect commonly used passwords, suggest some approaches that Morris may have used to construct the list. 10.5 What type of malware is the following code fragment? legitimate code if data is Friday the 13th; crash_computer(); legitimate code 10.6 Consider the following situation and identify the type of software attack, if any: You are the owner of a small business. After you login to your client server application with your credentials, you find that the data is displayed in the form of a jumbled collection of alphabets, numbers, special characters, and symbols. You are unpleasantly surprised and wonder what happened. You get a call after some time, and the person at the other end tells you that your system is hacked, and you can recover the data once you pay him a certain amount of money. 10.7 Assume that you have received an e-mail with an attachment from your friend's e-mail id. You access the e-mail using your work computer, and click on the 374 chapter 10 / Malicious Software attachment without screening it for

malware. What threats might this pose to your work computer? 10.8 Suppose you observe that your home PC is responding very slowly to information requests from the net. And then you further observe that your network gateway shows high levels of network activity, even though you have closed your e-mail client, Web browser, and other programs that access the net. What types of malware could cause these symptoms? Discuss how the malware might have gained access to your system. What steps can you take to check whether this has occurred? If you do identify malware on your PC, how can you restore it to safe operation? 10.9 Suppose while browsing the Internet, you get a popup window stating that you need to install this software in order to clean your system as it is running low on resources. Since the message seems to be from a genuine OS vendor like Microsoft Windows or Mac iOS, you click the 'OK' button. How could your action harm your system? How can you fix the issue? 10.10 Suppose you have a new smartphone and are excited about the range of apps available for it. You read about a really interesting new game that is available for your phone. You do a quick Web search for it and see that a version is available from one of the free marketplaces. When you download and start to install this app, you are asked to approve the access permissions granted to it. You see that it wants permission to "Send SMS messages" and to "Access your address-book." Should you be suspicious that a game wants these types of permissions? What threat might the app pose to your smartphone? Should you grant these permissions and proceed to install it? What types of malware might it be? 10.11 Assume you receive an e-mail that appears to come from a senior manager of your company, with a subject indicating that it concerns a project that you are currently working on. When you view the e-mail, you see that it asks you to review the attached revised press release, supplied as a PDF document, to check that all details are correct before management releases it. When you attempt to open the PDF, the viewer pops up a dialog labeled "Launch File," indicating that "the file and its viewer application are set to be launched by this PDF file." In the section of this dialog labeled "File" there are a number of blank lines and finally the text "Click the 'Open' button to view this document." You also note that there is a vertical scroll-bar visible for this region. What type of threat might this pose to your computer system should you indeed select the "Open" button? How could you check your suspicions without threatening your system? What type of attack is this type of message associated with? How many people are likely to have received this particular e-mail? 10.12 Assume you work in a financial auditing company. An e-mail arrives in your inbox that appears to be from your chief auditor with the following content: "We have identified a few threats which pose potential danger to our information systems. In order to address this, our information security team has decided to ensure proper credentials of all the employees. Please cooperate and complete this process immediately by clicking the given link." What kind of an attack is this e-mail attempting? How should you respond to such e-mails? 10.13 There are hundreds of unsolicited e-mails in your inbox. What kind of attack is this? Analyze related issues. 10.14 Suggest some methods of attacking the worm countermeasure architecture, discussed in Section 10.9, that could be used by worm creators. Suggest some possible countermeasures to these methods. 375 11.1 Intruders Intruder Behavior Patterns Intrusion Techniques 11.2 Intrusion Detection Audit Records Statistical Anomaly Detection Rule-Based Intrusion Detection The Base-Rate Fallacy Distributed Intrusion Detection Honeypots Intrusion Detection Exchange Format 11.3 Password Management The Vulnerability of Passwords The Use of Hashed Passwords User Password Choices Password Selection Strategies Bloom Filter 11.4 Key Terms, Review Questions, and Problems Chapter Intruders 376 chapter 11 / Intruders A significant security problem for

networked systems is hostile, or at least unwanted, trespass by users or software. User trespass can take the form of unauthorized logon to a machine or, in the case of an authorized user, acquisition of privileges or performance of actions beyond those that have been authorized. Software trespass can take the form of a virus, worm, or Trojan horse. All these attacks relate to network security because system entry can be achieved by means of a network. However, these attacks are not confined to network-based attacks. A user with access to a local terminal may attempt trespass without using an intermediate network. A virus or Trojan horse may be introduced into a system by means of an optical disc. Only the worm is a uniquely network phenomenon. Thus, system trespass is an area in which the concerns of network security and computer security overlap. Because the focus of this book is network security, we do not attempt a comprehensive analysis of either the attacks or the security countermeasures related to system trespass. Instead, in this Part we present a broad overview of these concerns. This chapter covers the subject of intruders. First, we examine the nature of the attack and then look at strategies intended for prevention and, failing that, detection. Next we examine the related topic of password management. 11.1 Intruders One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. In an important early study of intrusion, Anderson [ANDE80] identified three classes of intruders: ■■ Masquerader: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account Learning Objectives After studying this chapter, you should be able to: ◆ Distinguish among various types of intruder behavior patterns. ◆ Understand the basic principles of and requirements for intrusion detection. ◆ Discuss the key features of intrusion detection systems. ◆ Define the intrusion detection exchange format. ◆ Explain the purpose of honeypots. ◆ Explain the mechanism by which hashed passwords are used for user authentication. ◆ Understand the use of the Bloom filter in password management. 11.1 / Intruders 377 ■■ Misfeasor: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges ■■ Clandestine user: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection The masquerader is likely to be an outsider, the misfeasor generally is an insider, and the clandestine user can be either an outsider or an insider. Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. [GRAN04] lists the following examples of intrusion: ■■ Performing a remote root compromise of an e-mail server ■■ Defacing a Web server ■■ Guessing and cracking passwords ■■ Copying a database containing credit card numbers ■■ Viewing sensitive data, including payroll records and medical information, without authorization ■■ Running a packet sniffer on a workstation to capture usernames and passwords ■■ Using a permission error on an anonymous FTP server to distribute pirated software and music files ■■ Dialing into an unsecured modem and gaining internal network access ■■ Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password ■■ Using an unattended, logged-in workstation without permission Intruder Behavior Patterns The techniques and behavior patterns of intruders are constantly shifting, to exploit newly discovered weaknesses and to evade detection and countermeasures. Even so, intruders typically follow one of a number of recognizable behavior patterns, and these patterns typically differ from those

of ordinary users. In the following, we look at three broad examples of intruder behavior patterns, to give the reader some feel for the challenge facing the security administrator. Hackers Traditionally, those who hack into computers do so for the thrill of it or for status. The hacking community is a strong meritocracy in which status is determined by level of competence. Thus, attackers often look for targets of opportunity and then share the information with others. A typical example is a break-in at a large financial institution reported in [RADC04]. The intruder took advantage of the fact that the corporate network was running unprotected services, some of which were not even needed. In this case, the key to the break-in was the pcAnywhere application. The manufacturer, Symantec, advertises this program as a remote control 378 chapter 11 / Intruders solution that enables secure connection to remote devices. But the attacker had an easy time gaining access to pcAnywhere; the administrator used the same threeletter username and password for the program. In this case, there was no intrusion detection system on the 700-node corporate network. The intruder was only discovered when a vice-president walked into her office and saw the cursor moving files around on her Windows workstation. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign. Consequently, even for systems with no particularly sensitive resources, there is a motivation to control this problem. Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are designed to counter this type of hacker threat. In addition to using such systems, organizations can consider restricting remote logons to specific IP addresses and/or use virtual private network technology. One of the results of the growing awareness of the intruder problem has been the establishment of a number of computer emergency response teams (CERTs). These cooperative ventures collect information about system vulnerabilities and disseminate it to systems managers. Hackers also routinely read CERT reports. Thus, it is important for system administrators to quickly insert all software patches to discovered vulnerabilities. Unfortunately, given the complexity of many IT systems, and the rate at which patches are released, this is increasingly difficult to achieve without automated updating. Even then, there are problems caused by incompatibilities resulting from the updated software. Hence the need for multiple layers of defense in managing security threats to IT systems. Criminals Organized groups of hackers have become a widespread and common threat to Internet-based systems. These groups can be in the employ of a corporation or government but often are loosely affiliated gangs of hackers. Typically, these gangs are young, often Eastern European, Russian, or southeast Asian hackers who do business on the Web [ANTE06]. They meet in underground forums with names like DarkMarket.org and theftservices.com to trade tips and data and coordinate attacks. A common target is a credit card file at an e-commerce server. Attackers attempt to gain root access. The card numbers are used by organized crime gangs to purchase expensive items and are then posted to carder sites, where others can access and use the account numbers; this obscures usage patterns and complicates investigation. Whereas traditional hackers look for targets of opportunity, criminal hackers usually have specific targets, or at least classes of targets in mind. Once a site is penetrated, the attacker acts quickly, scooping up as much valuable information as possible and exiting. IDSs and IPSs can also be used for these types of attackers, but may be less effective because of the quick in-and-out nature of the attack. For e-commerce sites, database encryption should be used for sensitive customer information, especially credit cards. For hosted e-commerce sites (provided by an outsider service), the e-commerce organization should make use of a

dedicated server (not used to support multiple customers) and closely monitor the provider's security services. 11.1 / Intruders 379 Insider Attacks Insider attacks are among the most difficult to detect and prevent. Employees already have access and knowledge about the structure and content of corporate databases. Insider attacks can be motivated by revenge or simply a feeling of entitlement. An example of the former is the case of Kenneth Patterson, fired from his position as data communications manager for American Eagle Outfitters. Patterson disabled the company's ability to process credit card purchases during five days of the holiday season of 2002. As for a sense of entitlement, there have always been many employees who felt entitled to take extra office supplies for home use, but this now extends to corporate data. An example is that of a vice-president of sales for a stock analysis firm who quit to go to a competitor. Before she left, she copied the customer database to take with her. The offender reported feeling no animus toward her former employee; she simply wanted the data because it would be useful to her. Although IDS and IPS facilities can be useful in countering insider attacks, other more direct approaches are of higher priority. Examples include the following: ■■ Enforce least privilege, only allowing access to the resources employees need to do their job. ■■ Set logs to see what users access and what commands they are entering. ■■ Protect sensitive resources with strong authentication. ■■ Upon termination, delete employee's computer and network access. ■■ Upon termination, make a mirror image of employee's hard drive before reissuing it. That evidence might be needed if your company information turns up at a competitor. In this section, we look at the techniques used for intrusion. Then we examine ways to detect intrusion. Intrusion Techniques The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a backdoor into the system. Alternatively, the intruder attempts to acquire information that should have been protected. In some cases, this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways: ■■ One-way function: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible), in which the password is used to generate a key for the one-way function and in which a fixedlength output is produced. ■■ Access control: Access to the password file is limited to one or a very few accounts. 380 chapter 11 / Intruders If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers, [ALVA90] reports the following techniques for learning passwords: 1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults. 2. Exhaustively try all short passwords (those of one to three characters). 3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards. 4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies. 5. Try users' phone numbers, Social Security numbers, and room numbers. 6. Try all legitimate license plate numbers for this state. 7. Use a Trojan horse

(described in Chapter 10) to bypass restrictions on access. 8. Tap the line between a remote user and the host system. The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack. For example, a system can simply reject any login after three password attempts, thus requiring the intruder to reconnect to the host to try again. Under these circumstances, it is not practical to try more than a handful of passwords. However, the intruder is unlikely to try such crude methods. For example, if an intruder can gain access with a low level of privileges to an encrypted password file, then the strategy would be to capture that file and then use the encryption mechanism of that particular system at leisure until a valid password that provided greater privileges was discovered. Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable. Later in this chapter, we have much to say about thwarting guessing attacks. The seventh method of attack listed earlier, the Trojan horse, can be particularly difficult to counter. An example of a program that bypasses access controls has been cited in [ALVA90]. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background it also contained code to copy the password file, which was unencrypted but access protected, into the user's file. Because the game was running under the operator's high-privilege mode, it was able to gain access to the password file. The eighth attack listed, line tapping, is a matter of physical security. Other intrusion techniques do not require learning a password. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges. Privilege escalation can be done this way as well. We turn now to a discussion of the two principal countermeasures: detection and prevention. Detection is concerned with learning of an attack, either before or 11.2 / Intrusion Detection 381 after its success. Prevention is a challenging security goal and an uphill battle at all times. The difficulty stems from the fact that the defender must attempt to thwart all possible attacks, whereas the attacker is free to try to find the weakest link in the defense chain and attack at that point. 11.2 Intrusion Detection Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following: 1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved. 2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions. 3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility. Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. Of course, we cannot expect that there will be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user. Rather, we must expect that there will be some overlap. Figure 11.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of false

positives, or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection. In Anderson's study [ANDE80], it was postulated that one could, with reasonable confidence, distinguish between a masquerader and a legitimate user. Patterns of legitimate user behavior can be established by observing past history, and significant deviation from such patterns can be detected. Anderson suggests that the task of detecting a misfeasor (legitimate user performing in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behavior may be small. Anderson concluded that such violations would be undetectable solely through the search for anomalous behavior. However, misfeasor behavior might nevertheless be detectable by intelligent definition of the class of conditions that suggest unauthorized use. Finally, the detection of the clandestine user was felt to 382 chapter 11 / Intruders be beyond the scope of purely automated techniques. These observations, which were made in 1980, remain true today. [PORR92] identifies the following approaches to intrusion detection: 1. Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior. a. Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events. b. Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts. 2. Rule-based detection: Involves an attempt to define a set of rules or attack patterns that can be used to decide that a given behavior is that of an intruder. This is often referred to as signature detection. In essence, anomaly approaches attempt to define normal, or expected, behavior, whereas signature-based approaches attempt to define proper behavior. In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate. On the other hand, such techniques may be unable Figure 11.1 Profiles of Behavior of Intruders and Authorized Users Overlap in observed or expected behavior Prole of intruder behavior Prole of authorized user behavior Measurable behavior parameter Average behavior of intruder Average behavior of authorized user Probability density function 11.2 / Intrusion Detection 383 to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks. Audit Records A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used: ■■ Native audit records: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form. ■■ Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine. A good example of detection-specific audit records is one developed by Dorothy Denning

[DENN87]. Each audit record contains the following fields: ■■ Subject: A subject initiates actions. A subject could be a user or a process acting on behalf of users or groups of users. Subjects may be grouped into different access classes, and these classes may overlap. ■■ Action: An action initiated by a subject refers to some object; for example, login, read, perform I/O, execute. ■■ Object: Actions are performed on or with objects. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type. Object granularity may vary by object type and by environment. For example, database actions may be audited for the database as a whole or at the record level. ■■ Exception-Condition: If an exception condition occurs, this field contains identifying information. ■■ Resource-Usage: This is a list, in which each item gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time). ■■ Time-Stamp: The time stamp specifies the data and time of an action. Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command 384 chapter 11 / Intruders COPY GAME.EXE TO GAME.EXE issued by Smith to copy an executable file GAME from the current directory to the directory. The following audit records may be generated: Smith execute COPY.EXE 0 CPU = 00002 11058721678 Smith read GAME.EXE 0 RECORDS = 0 11058721679 Smith execute COPY.EXE write-viol RECORDS = 0 11058721680 In this case, the copy is aborted because Smith does not have write permission to . The decomposition of a user operation into elementary actions has three advantages: 1. Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access controls (by noting an abnormality in the number of exception conditions returned) and can detect successful subversions by noting an abnormality in the set of objects accessible to the subject. 2. Single-object, single-action audit records simplify the model and the implementation. 3. Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detectionspecific audit records. Statistical Anomaly Detection As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed. Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined. Because of the variability across users, such thresholds are likely to generate either a lot of false positives or a lot of false negatives. However, simple threshold detectors may be useful in conjunction with more sophisticated techniques. Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert. 11.2 / Intrusion Detection 385 The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of

audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior. Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior. Examples of metrics that are useful for profile-based intrusion detection are the following: ■■ Counter: A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute. ■■ Gauge: A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process. ■■ Interval timer: The length of time between two related events. An example is the length of time between successive logins to an account. ■■ Resource utilization: Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution. Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. [DENN87] lists the following approaches that may be taken: ■■ Mean and standard deviation ■■ Multivariate ■■ Markov process ■■ Time series ■■ Operational The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability. The use of mean and standard deviation is applicable to a wide variety of counters, timers, and resource measures. But these measures, by themselves, are typically too crude for intrusion detection purposes. The mean and standard deviation of a parameter are simple measures to calculate. Taken over a given period of time, these values provide a measure average behavior and its variability. These two calculations can be applied to a variety of counters, timers, and resource measures. However, these two measures are inadequate, by themselves, for effective intrusion detection. 386 chapter 11 / Intruders A multivariate calculation determines a correlate between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time). A Markov process estimates transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands. A time series model observes and calculates values based on a sequence of events over time. Such models can be used to detect a series of actions that happens to rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing. An operational model can be used to characterize what is considered abnormal, as opposed to performing an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits. This type of approach works best where intruder behavior can be deduced from certain types of activities. For example, a large number of login attempts over a short period suggests an attempted intrusion. As an example of the use of these various metrics and models, Table 11.1 shows various measures considered or tested for the Stanford Research Institute (SRI) Intrusion Detection System (IDES) [ANDE95, JAVI91] and the follow-on program Emerald [NEUM99]. The main advantage of the use of statistical profiles is that a prior knowledge of security flaws is not required. The detector program learns what is “normal” behavior and then looks for deviations. The approach is not based on systemdependent

characteristics and vulnerabilities. Thus, it should be readily portable among a variety of systems. Rule-Based Intrusion Detection Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious. In very general terms, we can characterize all approaches as focusing on either anomaly detection or penetration identification, although there is some overlap in these approaches. Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to automatically generate rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior. As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past. In order for this approach to be effective, a rather large database of rules will be needed. For example, a scheme described in [VACC89] contains anywhere from 104 to 106 rules. 11.2 / Intrusion Detection 387 Rule-based penetration identification takes a very different approach to intrusion detection. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules used in these systems are specific to the machine and operating system. The most fruitful approach to developing such rules is to analyze attack tools and scripts collected on the Internet. These rules can be supplemented with rules generated by knowledgeable security personnel. In this latter case, the normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the target system. Measure Model Type of Intrusion Detected Login and Session Activity Login frequency by day and time Mean and standard deviation Intruders may be likely to log in during off-hours Frequency of login at different locations Mean and standard deviation Intruders may log in from a location that a particular user rarely or never uses Time since last login Operational Break in on a “dead” account Elapsed time per session Mean and standard deviation Significant deviations might indicate masquerader Quantity of output to location Mean and standard deviation Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data Session resource utilization Mean and standard deviation Unusual processor or I/O levels could signal an intruder Password failures at login Operational Attempted break-in by password guessing Failures to login from specified terminals Operational Attempted break-in Command or Program Execution Activity Execution frequency Mean and standard deviation May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands Program resource utilization Mean and standard deviation An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization Execution denials Operational model May detect penetration attempt by individual user who seeks higher privileges File Access Activity Read, write, create, delete frequency Mean and standard deviation Abnormalities for read and write access for individual users may signify masquerading or browsing Records read, written Mean and standard deviation Abnormality could signify an attempt

to obtain sensitive data by inference and aggregation Failure count for read, write, create, delete Operational May detect users who persistently attempt to access unauthorized files Table 11.1 Measures That May Be Used for Intrusion Detection 388 chapter 11 / Intruders A simple example of the type of rules that can be used is found in NIDX, an early system that used heuristic rules that can be used to assign degrees of suspicion to activities [BAUE88]. Example heuristics are the following: 1. Suspicious activity: A user accesses the personal directory of another user and attempts to read files in that directory. 2. Suspicious activity: A user accesses the personal directory of another user and attempts to write or create files in that directory. 3. Expected activity: A user logs in after hours and accesses the same file he or she accessed during business hours. 4. Suspicious activity: A user opens a disk devices directly rather than relying on higher-level operating system utilities. 5. Suspicious activity: A user is logged onto one system twice at the same time. 6. Suspicious activity: A user makes copies of system programs. The penetration identification scheme used in IDES is representative of the strategy followed. Audit records are examined as they are generated, and they are matched against the rule base. If a match is found, then the user's suspicion rating is increased. If enough rules are matched, then the rating will pass a threshold that results in the reporting of an anomaly. The IDES approach is based on an examination of audit records. A weakness of this plan is its lack of flexibility. For a given penetration scenario, there may be a number of alternative audit record sequences that could be produced, each varying from the others slightly or in subtle ways. It may be difficult to pin down all these variations in explicit rules. Another method is to develop a higher-level model independent of specific audit records. An example of this is a state transition model known as USTAT [VIGN02, ILGU95]. USTAT deals in general actions rather than the detailed specific actions recorded by the UNIX auditing mechanism. USTAT is implemented on a SunOS system that provides audit records on 239 events. Of these, only 28 are used by a preprocessor, which maps these onto 10 general actions (Table 11.2). Using just these actions and the parameters that are invoked with each action, a state transition diagram is developed that characterizes suspicious activity. Because a number of different auditable events map into a smaller number of actions, the rule-creation process is simpler. Furthermore, the state transition diagram model is easily modified to accommodate newly learned intrusion behaviors. The Base-Rate Fallacy To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms or much time will be wasted analyzing the false alarms. Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the 11.2 / Intrusion Detection 389 number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating. This is an example of a phenomenon known as the base-rate fallacy. A study of existing intrusion detection systems, reported in [AXEL00], indicated that current systems have not overcome the problem of the base-rate fallacy. See Appendix J for a brief background on the mathematics of this problem. Distributed Intrusion Detection Traditionally, work on intrusion detection systems focused on single-system standalone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. Although it is possible to mount a defense by using stand-alone

intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network. Porras points out the following major issues in the design of a distributed intrusion detection system [PORR92]: ■■ A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records. ■■ One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable. USTAT Action SunOS Event Type Read open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt Write truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct Create mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod Delete rmdir, unlink Execute exec, execve Exit exit Modify_Owner chown, fchown Modify_Perm chmod, fchmod Rename rename Hardlink link Table 11.2 USTAT Actions Versus SunOS Event Types 390 chapter 11 / Intruders ■■ Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all audit data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there are more than one analysis centers, but these must coordinate their activities and exchange information. A good example of a distributed intrusion detection system is one developed at the University of California at Davis [HEBE92, SNAP91]. A similar approach has been taken for a project at Purdue [SPAF00, BALA98]. Figure 11.2 shows the overall architecture, which consists of three main components: ■■ Host agent module: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on securityrelated events on the host and transmit these to the central manager. ■■ LAN monitor agent module: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager. ■■ Central manager module: Receives reports from LAN monitor and host agents, and processes and correlates these reports to detect intrusion. The scheme is designed to be independent of any operating system or system auditing implementation. Figure 11.3 shows the general approach that is taken. The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest. These records are then reformatted into a standardized format referred to as the host audit Figure 11.2 Architecture for Distributed Intrusion Detection Central manager LAN monitor Host Host Agent module Router WAN Manager module 11.2 / Intrusion Detection 391 record (HAR). Next, a template-driven logic module analyzes the records for suspicious activity. At the lowest level, the agent scans for notable events that are of interest independent of any past events. Examples include failed file accesses, accessing system files, and changing a file's access control. At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures). Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like. When suspicious activity is detected, an alert is sent to the central manager. The central

manager includes an expert system that can draw inferences from received data. The manager may also query individual systems for copies of HARs to correlate with those from other agents. The LAN monitor agent also supplies information to the central manager. The LAN monitor agent audits host-host connections, services used, and volume of traffic. It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin. The architecture depicted in Figures 11.2 and 11.3 is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected. Honeypots A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to ■■ divert an attacker from accessing critical systems ■■ collect information about the attacker's activity ■■ encourage the attacker to stay on the system long enough for administrators to respond Figure 11.3 Agent Architecture OS audit information Alerts Modi cations Query/ response Notable activity; Signatures; Noteworthy sessions Host audit record (HAR) Filter for security interest Reformat function OS audit function Analysis module Templates Central manager Logic module 392 chapter 11 / Intruders These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect. The system is instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker's activities. Because any attack against the honeypot is made to seem successful, administrators have time to mobilize and log and track the attacker without ever exposing productive systems. The honeypot is a resource that has no production value. There is no legitimate reason for anyone outside the network to interact with a honeypot. Thus, any attempt to communicate with the system is most likely a probe, scan, or attack. Conversely, if a honeypot initiates outbound communication, the system has probably been compromised. Initial efforts involved a single honeypot computer with IP addresses designed to attract hackers. More recent research has focused on building entire honeypot networks that emulate an enterprise, possibly with actual or simulated traffic and data. Once hackers are within the network, administrators can observe their behavior in detail and figure out defenses. Honeypots can be deployed in a variety of locations. Figure 11.4 illustrates some possibilities. The location depends on a number of factors, such as the type of information the organization is interested in gathering and the level of risk that organizations can tolerate to obtain the maximum amount of data. A honeypot outside the external firewall (location 1) is useful for tracking attempts to connect to unused IP addresses within the scope of the network. A honeypot at this location does not increase the risk for the internal network. The danger of having a compromised system behind the firewall is avoided. Further, because the honeypot attracts many potential attacks, it reduces the alerts issued by the firewall and by internal IDS sensors, easing the management burden. The disadvantage of an external honeypot is that it has little or no ability to trap internal attackers, especially if the external firewall filters traffic in both directions. The network of externally available services, such as Web and mail, often called the DMZ (demilitarized zone), is another candidate for locating a honeypot (location 2). The security administrator must assure that the other systems in the DMZ are secure against any activity generated by the honeypot. A disadvantage of this location is that a typical DMZ is not fully accessible, and the firewall typically blocks traffic to the DMZ that

attempts to access unneeded services. Thus, the firewall either has to open up the traffic beyond what is permissible, which is risky, or limit the effectiveness of the honeypot. A fully internal honeypot (location 3) has several advantages. Its most important advantage is that it can catch internal attacks. A honeypot at this location can also detect a misconfigured firewall that forwards impermissible traffic from the Internet to the internal network. There are several disadvantages. The most serious of these is if the honeypot is compromised so that it can attack other internal systems. Any further traffic from the Internet to the attacker is not blocked by the firewall because it is regarded as traffic to the honeypot only. Another difficulty for this honeypot location is that, as with location 2, the firewall must adjust its filtering to allow traffic to the honeypot, thus complicating firewall configuration and potentially compromising the internal network. 11.2 / Intrusion Detection 393 Intrusion Detection Exchange Format To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them. The working group issued the following RFCs in 2007: ■■ Intrusion Detection Message Exchange Requirements (RFC 4766): This document defines requirements for the Intrusion Detection Message Exchange Format (IDMEF). The document also specifies requirements for a communication protocol for communicating IDMEF. Figure 11.4 Example of Honeypot Deployment Internet External rewall Honeypot Honeypot Honeypot LAN switch or router LAN switch or router Internal network Service network (Web, Mail, DNS, etc.) 2 1 3 394 chapter 11 / Intruders ■■ The Intrusion Detection Message Exchange Format (RFC 4765): This document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented, an XML Document Type Definition is developed, and examples are provided. ■■ The Intrusion Detection Exchange Protocol (RFC 4767): This document describes the Intrusion Detection Exchange Protocol (IDXP), an applicationlevel protocol for exchanging data between intrusion detection entities. IDXP supports mutual authentication, integrity, and confidentiality over a connection-oriented protocol. Figure 11.5 illustrates the key elements of the model on which the intrusion detection message exchange approach is based. This model does not correspond to any particular product or implementation, but its functional components are the key elements of any IDS. The functional components are as follows: Figure 11.5 Model for Intrusion Detection Message Exchange Response Activity Event Event Alert Noti cation Operator Administrator Security policy Security policy 11.2 / Intrusion Detection 395 ■■ Data source: The raw data that an IDS uses to detect unauthorized or undesired activity. Common data sources include network packets, operating system audit logs, application audit logs, and system-generated checksum data. ■■ Sensor: Collects data from the data source. The sensor forwards events to the analyzer. ■■ Analyzer: The ID component or process that analyzes the data collected by the sensor for signs of unauthorized or undesired activity or for events that might be of interest to the security administrator. In many existing IDSs, the sensor and the analyzer are part of the same component. ■■ Administrator: The human with overall responsibility for setting the security policy of the organization, and, thus, for decisions about deploying and configuring the IDS. This may or may not be the same person as the operator of the IDS. In some organizations, the administrator is associated with the

network or systems administration groups. In other organizations, it's an independent position. ■■ Manager: The ID component or process from which the operator manages the various components of the ID system. Management functions typically include sensor configuration, analyzer configuration, event notification management, data consolidation, and reporting. ■■ Operator: The human that is the primary user of the IDS manager. The operator often monitors the output of the IDS and initiates or recommends further action. In this model, intrusion detection proceeds in the following manner. The sensor monitors data sources looking for suspicious activity, such as network sessions showing unexpected telnet activity, operating system log file entries showing a user attempting to access files to which he or she is not authorized to have access, and application log files showing persistent login failures. The sensor communicates suspicious activity to the analyzer as an event, which characterizes an activity within a given period of time. If the analyzer determines that the event is of interest, it sends an alert to the manager component that contains information about the unusual activity that was detected, as well as the specifics of the occurrence. The manager component issues a notification to the human operator. A response can be initiated automatically by the manager component or by the human operator. Examples of responses include logging the activity; recording the raw data (from the data source) that characterized the event; terminating a network, user, or application session; or altering network or system access controls. The security policy is the predefined, formally documented statement that defines what activities are allowed to take place on an organization's network or on particular hosts to support the organization's requirements. This includes, but is not limited to, which hosts are to be denied external network access. The specification defines formats for event and alert messages, message types, and exchange protocols for communication of intrusion detection information. 396 chapter 11 / Intruders 11.3 Password Management The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways: ■■ The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access. ■■ The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others. ■■ The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user. The Vulnerability of Passwords In this subsection, we outline the main forms of attack against password-based authentication and briefly outline a countermeasure strategy. The remainder of Section 11.3 goes into more detail on the key countermeasures. Typically, a system that uses password-based authentication maintains a password file indexed by user ID. One technique that is typically used is to store not the user's password but a one-way hash function of the password, as described subsequently. We can identify the following attack strategies and countermeasures: ■■ Offline dictionary attack: Typically, strong access controls are used to protect the system's password file. However, experience shows that determined hackers can frequently bypass such controls and gain access to the file. The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by

that ID/password combination. Countermeasures include controls to prevent unauthorized access to the password file, intrusion detection measures to identify a compromise, and rapid reissuance of passwords should the password file be compromised. ■■ Specific account attack: The attacker targets a specific account and submits password guesses until the correct password is discovered. The standard countermeasure is an account lockout mechanism, which locks out access to the account after a number of failed login attempts. Typical practice is no more than five access attempts. ■■ Popular password attack: A variation of the preceding attack is to use a popular password and try it against a wide range of user IDs. A user's tendency is to choose a password that is easily remembered; this unfortunately makes the 11.3 / Password Management 397 password easy to guess. Countermeasures include policies to inhibit the selection by users of common passwords and scanning the IP addresses of authentication requests and client cookies for submission patterns. ■■ Password guessing against single user: The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password. Countermeasures include training in and enforcement of password policies that make passwords difficult to guess. Such policies address the secrecy, minimum length of the password, character set, prohibition against using well-known user identifiers, and length of time before the password must be changed. ■■ Workstation hijacking: The attacker waits until a logged-in workstation is unattended. The standard countermeasure is automatically logging the workstation out after a period of inactivity. Intrusion detection schemes can be used to detect changes in user behavior. ■■ Exploiting user mistakes: If the system assigns a password, then the user is more likely to write it down because it is difficult to remember. This situation creates the potential for an adversary to read the written password. A user may intentionally share a password, to enable a colleague to share files, for example. Also, attackers are frequently successful in obtaining passwords by using social engineering tactics that trick the user or an account manager into revealing a password. Many computer systems are shipped with preconfigured passwords for system administrators. Unless these preconfigured passwords are changed, they are easily guessed. Countermeasures include user training, intrusion detection, and simpler passwords combined with another authentication mechanism. ■■ Exploiting multiple password use: Attacks can also become much more effective or damaging if different network devices share the same or a similar password for a given user. Countermeasures include a policy that forbids the same or similar password on particular network devices. ■■ Electronic monitoring: If a password is communicated across a network to log on to a remote system, it is vulnerable to eavesdropping. Simple encryption will not fix this problem, because the encrypted password is, in effect, the password and can be observed and reused by an adversary. The Use of Hashed Passwords A widely used password security technique is the use of hashed passwords and a salt value. This scheme is found on virtually all UNIX variants as well as on a number of other operating systems. The following procedure is employed (Figure 11.6a). To load a new password into the system, the user selects or is assigned a password. This password is combined with a fixed-length salt value [MORR79]. In older implementations, this value is related to the time at which the password is assigned to the user. Newer implementations use a pseudorandom or random number. The password and salt serve as inputs to a hashing algorithm to produce a fixed-length hash code. The hash algorithm is designed to be slow to execute to thwart attacks. The hashed password is then stored, together with a plaintext copy of the salt, in 398 chapter 11 / Intruders the password file for the corresponding user ID. The hashed-

password method has been shown to be secure against a variety of cryptanalytic attacks [WAGN00]. When a user attempts to log on to a UNIX system, the user provides an ID and a password (Figure 11.6b). The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted. Figure 11.6 UNIX Password Scheme User ID Salt Password Load Select (a) Loading a new password (b) Verifying a password Salt • • • Password le Hash code User ID User ID Salt Password le Slow hash function Salt Hashed password Password Slow hash function Compare Hash code M1 11.3 / Password Management 399 The salt serves three purposes: ■■ It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned different salt values. Hence, the hashed passwords of the two users will differ. ■■ It greatly increases the difficulty of offline dictionary attacks. For a salt of length b bits, the number of possible passwords is increased by a factor of 2b , increasing the difficulty of guessing a password in a dictionary attack. ■■ It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them. To see the second point, consider the way that an offline dictionary attack would work. The attacker obtains a copy of the password file. Suppose first that the salt is not used. The attacker's goal is to guess a single password. To that end, the attacker submits a large number of likely passwords to the hashing function. If any of the guesses matches one of the hashes in the file, then the attacker has found a password that is in the file. But faced with the UNIX scheme, the attacker must take each guess and submit it to the hash function once for each salt value in the dictionary file, multiplying the number of guesses that must be checked. There are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine. The attacker should be able to check many thousands of possible passwords with little resource consumption. In addition, if an opponent is able to obtain a copy of the password file, then a cracker program can be run on another machine at leisure. This enables the opponent to run through millions of possible passwords in a reasonable period. UNIX Implementations Since the original development of UNIX, most implementations have relied on the following password scheme. Each user selects a password of up to eight printable characters in length. This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The hash routine, known as crypt(3), is based on DES. A 12-bit salt value is used. The modified DES algorithm is executed with a data input consisting of a 64-bit block of zeros. The output of the algorithm then serves as input for a second encryption. This process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into an 11-character sequence. The modification of the DES algorithm converts it into a one-way hash function. The crypt(3) routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25. This particular implementation is now considered woefully inadequate. For example, [PERR03] reports the results of a dictionary attack using a supercomputer. The attack was able to process over 50 million password guesses in about 80 minutes. Further, the results showed that for about $10,000 anyone should be able to do the same in a few months using one uniprocessor machine. Despite its known weaknesses, this UNIX scheme is still often required for compatibility with existing account management software or in multivendor

environments. 400 chapter 11 / Intruders There are other, much stronger, hash/salt schemes available for UNIX. The recommended hash function for many UNIX systems, including Linux, Solaris, and FreeBSD (a widely used open source UNIX implementation), is based on the MD5 secure hash algorithm (which is similar to, but not as secure as SHA-1). The MD5 crypt routine uses a salt of up to 48 bits and effectively has no limitations on password length. It produces a 128-bit hash value. It is also far slower than crypt(3). To achieve the slowdown, MD5 crypt uses an inner loop with 1000 iterations. Probably the most secure version of the UNIX hash/salt scheme was developed for OpenBSD, another widely used open source UNIX. This scheme, reported in [PROV99], uses a hash function based on the Blowfish symmetric block cipher. The hash function, called Bcrypt, is quite slow to execute. Bcrypt allows passwords of up to 55 characters in length and requires a random salt value of 128 bits, to produce a 192-bit hash value. Bcrypt also includes a cost variable; an increase in the cost variable causes a corresponding increase in the time required to perform a Bcyrpt hash. The cost assigned to a new password is configurable, so that administrators can assign a higher cost to privileged users. Password Cracking Approaches The traditional approach to password guessing, or password cracking as it is called, is to develop a large dictionary of possible passwords and to try each of these against the password file. This means that each password must be hashed using each available salt value and then compared to stored hash values. If no match is found, then the cracking program tries variations on all the words in its dictionary of likely passwords. Such variations include backwards spelling of words, additional numbers or special characters, or sequence of characters, An alternative is to trade off space for time by precomputing potential hash values. In this approach the attacker generates a large dictionary of possible passwords. For each password, the attacker generates the hash values associated with each possible salt value. The result is a mammoth table of hash values known as a rainbow table. For example, [OECH03] showed that using 1.4 GB of data, he could crack 99.9% of all alphanumeric Windows password hashes in 13.8 seconds. This approach can be countered by using a sufficiently large salt value and a sufficiently large hash length. Both the FreeBSD and OpenBSD approaches should be secure from this attack for the foreseeable future. User Password Choices Even the stupendous guessing rates referenced in the preceding section do not yet make it feasible for an attacker to use a dumb brute-force technique of trying all possible combinations of characters to discover a password. Instead, password crackers rely on the fact that some people choose easily guessable passwords. Some users, when permitted to choose their own password, pick one that is absurdly short. One study at Purdue University [SPAF92a] observed password change choices on 54 machines, representing approximately 7000 user accounts. Almost 3% of the passwords were three characters or fewer in length. An attacker could begin the attack by exhaustively testing all possible passwords of length 3 or 11.3 / Password Management 401 fewer. A simple remedy is for the system to reject any password choice of fewer than, say, six characters or even to require that all passwords be exactly eight characters in length. Most users would not complain about such a restriction. Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward. The cracker simply has to test the password file against lists of likely passwords. Because many people use guessable passwords, such a strategy should succeed on virtually all systems. One demonstration of the effectiveness of guessing is reported in [KLEI90]. From a variety of sources, the author collected UNIX