CHAPTER

Operating System Security 11

CHAPTER OUTLINE

INFORMATION IN THIS CHAPTER

- Operating system hardening
- Protecting against malware

The Basics of Information Security. DOI: http://dx.doi.org/10.1016/B978-0-12-800744-0.00011-7

- Software firewalls and host intrusion detection
- Operating system security tools

INTRODUCTION

When we seek to protect our data, processes, and applications against concerted attacks, one of the largest areas in which we find weaknesses is on the operating system that hosts all of these (be it a computer, router, or smartphone). If we do not take care to protect our operating systems, we really have no basis for getting to a reasonably strong security posture.

There are a number of ways by which we can mitigate the various threats and vulnerabilities we might face from an operating system perspective. One of the easiest categories we can point out is operating system hardening. We can use this technique when we are configuring hosts that might face hostile action in order to decrease the number of openings through which an attacker might ultimately reach us.

We can also add tools and applications to our operating system that are designed to combat some of the techniques attackers might use against us. The most common and obvious of these is the use of anti-malware tools, which we will discuss later in this chapter, that protect us from the broad variety of malicious code to which our system might be exposed, particularly if it is Internet facing. In the same general class of software, we can also look to software firewalls and host-based intrusion detection systems (HIDS) in order to block unwanted traffic and to alert us when undesirable network traffic is arriving at, or originating from, our systems.

Additionally we can use the large number of security tools that are available to help us detect potentially vulnerable areas on our hosts. We might use such tools to find services that we did not discover during our hardening effort, locate network services that are known to contain exploitable flaws, validate our patching is up to date and generally inspect/audit our systems.

Through the combination of all these efforts, once again to return to the concept of defense in depth, we can mitigate many of the security issues we might find on the hosts for which we are responsible.

Operating system hardening

When we look at operating system hardening, we arrive at a new concept in information security. One of the main goals of operating system hardening is to reduce the number of available avenues through which our operating system might be attacked. The total of these areas is referred to as our attack surface [1]. The larger our attack surface is, the greater chance we stand of an attacker

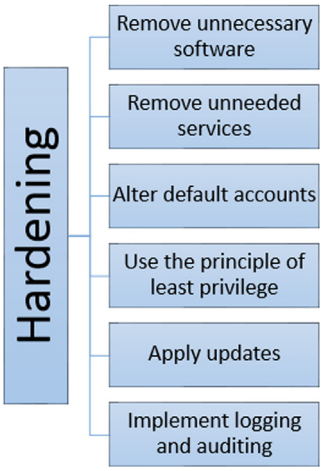

FIGURE 11.1

Six main hardening categories.

successfully penetrating our defenses. Each area in which we are potentially insecure adds to our attack surface, and each area in which we have applied security measures decreases it.

There are six main ways in which we can decrease our attack surface, as listed here and shown in Figure 11.1:

1. Removing unnecessary software
2. Removing or turning off unessential services
3. Making alterations to common accounts
4. Applying the principle of least privilege
5. Applying software updates in a timely manner
6. Making use of logging and auditing functions

Remove all unnecessary software

Alert!

We should always exercise great care when making changes to operating system settings, tools, and software. Some of the changes we might make could have unintended effects on the way our operating system functions, and a production machine is not the place to learn this through experience. Researching changes carefully and testing on lab systems before we make changes is always a good policy.

Each piece of software installed on our operating system adds to our attack surface. Some software may have a much greater effect than others, but they all add up. If we are truly seeking to harden our operating system, we need to take a

hard look at the software that should be loaded on it, and take steps to ensure that we are working with the bare minimum need for a functional system.

If we are preparing a Web server, for instance, we should have the Web server software, any libraries or code interpreters that are needed to support the Web server, and any utilities that deal with the administration and maintenance of the operating system, such as backup software and remote access tools. We should remove applications like Microsoft office or services like File Transfer Protocol (FTP). We really have no reason to install anything else if the system is truly going to function solely as a Web server.

Our problems begin to arise when we see other software installed on the machine, often with the best of intentions. For example, let us say that one of our developers logs in remotely and needs to make a change to a Web page on the fly, so they install the Web development software they need. Then they need to evaluate the changes, so they install their favorite Web browser and the associated media plug-ins, such as Adobe Flash and Acrobat Reader, as well as a video player to test the video content. In very short order, not only do we have software that should not be there, but the software quickly becomes outdated since it is not patched, because it is not "officially" installed so is not part of the configuration management plan. At this point, we have a relatively serious security issue on an Internet-facing machine.

Remove all unessential services

In the same vein as removing unneeded software, we should also remove or disable unessential services. Many operating systems ship with a wide variety of services turned on in order to share information over the network, locate other devices, synchronize the time, allow files to be accessed and transferred, and perform other tasks. We may also find that services have been installed by various applications, to provide the tools and resources on which the application depends in order to function.

Turning operating services off can be an exercise in experimentation and frustration. In many cases, such services are not named in a fashion that indicates their actual function, and tracking down what each of them is doing may require a bit of research. One of the best things to do first when we are seeking to locate such extraneous services is to determine the network ports on which the system is actually listening for network connections. Many operating systems have built-in utilities that will allow us to do this, such as netstat on Microsoft operating systems, but we can also put Nmap to use for such tasks.

As we discussed in Chapter 10, Nmap can allow us to discover the devices on our networks, but it can also allow us to determine on which network ports a given system is listening. If we run the following Nmap command:

Nmap <IP address>

we will see results similar to those shown in Figure 11.2.

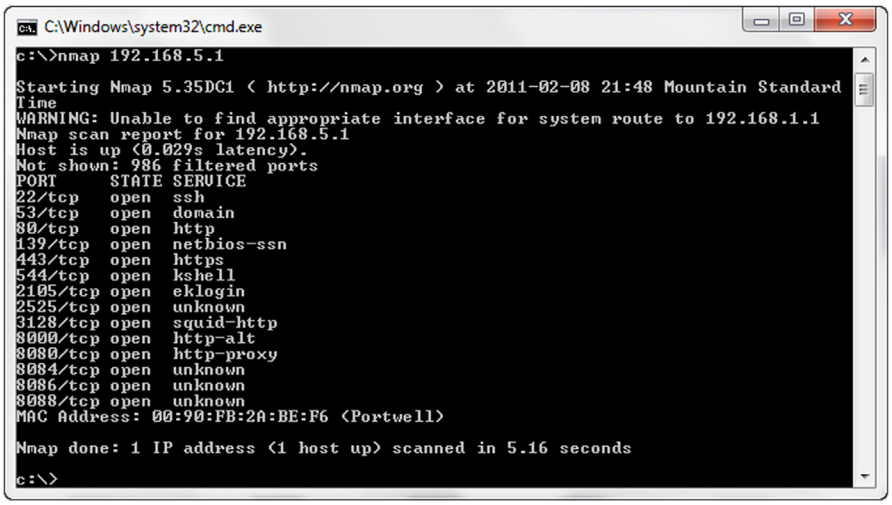

FIGURE 11.2

Nmap scan result.

In this case, we can immediately point out several common services running on the target:

- **Port 22** Remote access to the system, secured with Secure Shell (SSH)
- **Port 53** Domain name system (DNS), which translates friendly names to IP addresses
- **Port 80** Hypertext Transfer Protocol (HTTP), which serves Web content
- **Port 443** Hypertext Transfer Protocol Secure (HTTPS), which serves Web pages secured with Secure Sockets Layer (SSL) and/or Transport Layer Security (TLS)

Several other ports are open as well, running various services. We can use this information as a starting place for closing down undesirable services. In the case of our example target, ports 22, 80, and 443 being open might be notable if we did not intend to allow remote access or serve Web content.

Alter default accounts

A common weakness in many operating systems is the use of accounts known to be standard. In many operating systems (as well as some applications), we can find the equivalent of a guest account and an administrator account. We may also find a variety of others, including those intended for the use of support personnel, to allow services or utilities to operate, and a plethora of others, widely varying by the operating system vendor, version, and so forth. Such accounts are commonly referred to as default accounts.

In some cases, the default accounts may come equipped with excessively liberal permissions to regulate the actions they are allowed to carry out, which can cause a great deal of trouble when they are being used by an informed attacker. We may also find that default accounts are set with a particular password or no password at all. If we allow such accounts to remain on the system with their default settings, we may be leaving the proverbial doors that protect access to our system wide open so that attackers can simply stroll right in and make themselves at home.

Typical measures we would take to mitigate such security risks are generally very simple to carry out. We should first decide whether the accounts are needed at all, and disable or remove any we will not be using. In the case of guest accounts, support accounts, and others of a similar nature, we can often quickly and easily turn the accounts off or remove them entirely without causing problems for ourselves. In the case of administrative accounts, often with names such as administrator, admin, or root, we may not be able to safely remove them from the system, or the operating system may prevent us from doing so. In most cases, however, such accounts can be renamed in order to confound attackers who might attempt to make use of them. Lastly, we should not leave any account with a default password, no matter what its status; as such passwords are often documented and well known.

Apply the principle of least privilege

As we discussed in Chapter 3, the principle of least privilege dictates that we only allow a party the absolute minimum permission needed for it to carry out its function. Depending on the operating system in question, we may find this idea put into practice to a greater or a lesser extent. In almost any modern operating system, we can find the tasks a particular user is allowed to carry out separated into those that require administrative privileges and those that do not.

In general, normal operating system users are allowed to read and write files, and perhaps execute scripts or programs, but they are limited to doing so within a certain restricted portion of the file system. Normal users are generally not allowed to carry out tasks such as modifying the way hardware functions, making changes to the files on which the operating system itself depends, and installing software that can change or affect the entire operating system. Such activities are generally restricted to those users that are allowed administrative access.

On most UNIX and Linux-like operating systems, we can often see such roles strictly enforced. Although it would be possible for the administrator of such a system to allow all users to act with the privileges of an administrator, this is generally not the convention and administrative or “root” access is often guarded carefully. On Microsoft operating systems, we can often find the exact opposite to be true. On a windows system the default is to give users more control, so care needs to be taken to change permissions to be more restrictive. While there are more threats focused on MS due to the fact they have larger market share, the security posture for any system is based on the administrator. The same paradigm exists between Apple IOS and Android IOS in the smartphone market.

When we allow the average system user to regularly function with administrative privileges, we leave ourselves open to a wide array of security issues. If the user executes a malware-infected file or application, he does so as the administrator and that program has considerably more freedom to alter the operating system and other software installed on the host. If an attacker compromises a user's account, and that account has been given administrative rights, we have now given the keys to the entire system directly to the attacker. Nearly any type of attack we might discuss, launched from nearly any source, will have considerably more impact when allowed access to administrative rights on a host. Thus one of the first actions a hacker will take if they break in via a user account is privilege escalation. It is important to monitor admin accounts for misuse!

If, instead, we limit the privileges on our systems to the minimum needed in order to allow our users to perform their required tasks, we go a long way toward mitigating many security issues. In many cases, attacks will fail entirely when an attacker attempts to run them from a user account running with a limited set of permissions. This is a very cheap security measure we can put in place and is simple to implement. Many users will complain about the inability to install new software, so it is key to have policy supporting this practice and ensure users understand the reason for the policy.

Perform updates

Regular and timely updates to our operating systems and applications are critical to maintaining strong security. New attacks are published on a regular basis, and if we do not apply the security patches released by the vendors that manufacture our operating systems and applications, we will likely fall victim very quickly to a large number of well-known attacks.

We can look to the various items of malware propagating over the Internet at any given time as an excellent example of this idea. Many pieces of malware are able to spread by exploiting known vulnerabilities that have long since been patched by the software vendors. Although it does pay to be prudent when planning to install software updates and to test them thoroughly before doing so, it is generally unwise to delay this process for very long.

One of the most crucial times to ensure that we have properly patched a system is directly after we have finished installing it. If we connect a newly installed and completely unpatched system to our network, we may see it attacked and compromised in very short order, even on internal networks. The commonly considered best practice in such a situation is to download the patches onto removable media and use this media to patch the system before ever connecting it to a network. Part of solid configuration management program is to monitor patch announcements. There are services that will do this for you. You must also consider auto patching for systems like your home computer. Patching is one of the most important parts of your security program (even it is part of the IT department function).

Turn on logging and auditing

Last, but certainly not least, we should configure and turn on the appropriate logging and auditing features for our system. Although the particulars of how we configure such services may vary slightly depending on the operating system in question, and the use to which the system is to be put, we generally need to be able to keep an accurate and complete record of the important processes and activities that take place on our systems. We will generally want to log significant events such as the exercise of administrative privileges, users logging in to and out of the system, or failing to log in, changes made to the operating system, and a number of similar activities taking place. For a simple Windows OS, there are over 200 security-related logs that can be turned on so it is important to find the right balance of logs and storage. Key logs should be tied to alerts and a monitoring program.

Depending on the environment into which we will be placing the system, we may also want to include additional features to supplement the tools built into the operating system for these purposes. We may want to install a variety of monitoring tools that watch the functionality of the system and alert us to issues with the system itself or anomalies that might show in the various system or application logs. We might also want to install supplementary logging architecture in order to monitor the activities of multiple machines or to simply allow duplicate remote copies of logs to be maintained outside the system to help ensure that we have an unaltered record of the activities that might have taken place on the system.

> In addition to the hardening methods discussed above, there are a number of specific hardening standards. Some of the more commonly discussed are the Security Technical Implementation Guides (STIGs)[1] from the US Defense Information Systems Agency (DISA) and the hardening guidelines[2] available from the US National Security Agency (NSA)

It is also important to note that while logs are key to a post event investigation, actually reviewing the logs is a vital part of the process. If we collect logs but never review them we will miss detecting attacks early and suffer much greater overall impact.

Protecting against malware

A large concern at present is the mind-boggling number and variety of malware present on the networks, systems, and storage devices around the globe. Using such tools, attackers can disable systems, steal data, conduct social engineering

[1] http://iase.disa.mil/stigs/.
[2] http://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/operating_systems.shtml.

attacks, blackmail users, gather intelligence, and perform a number of other attacks.

A good example of a particularly complex and impactful item of malware to examine is Stuxnet. Stuxnet was first discovered in July 2009, albeit in a somewhat weaker form than what it ultimately reached [2]. Although the number of systems infected with it was much lower in comparison to some of the other major malware outbreaks that have taken place over the years, it had a much more specific focus in that it targeted the Supervisory Control and Data Acquisition (SCADA) systems that run various industrial processes. In the case of this attack, it was a nation state attacking another nation state's military capability.

Additional resources

For considerably more details on what Stuxnet does and how it does it, see the "W32.Stuxnet Dossier" from Symantec, available at www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

The ultimate goal of Stuxnet appears to have been the sabotage of SCADA systems, largely targeted at portions of the equipment running in the nuclear program in Iran [3]. Stuxnet has raised the bar for malware from largely being a virtual-based attack to actually being physically destructive.

Anti-malware tools

Most anti-malware applications detect threats in the same way the IDS we discussed in Chapter 10 do: either by matching against a signature or by detecting anomalous activities taking place. Anti-malware tools do tend to depend more heavily on signatures than on anomaly detection, which is typically referred to in the anti-malware field as heuristics. Malware signatures are usually updated by the vendor of the application at least once a day and may be updated more often than that if the need arises.

Anti-malware tools generally detect malware in one of two main ways: either by detecting the presence of, or traffic indicative of, malware in real time or by performing scans of the files and processes already in place on the system. When malware is found, responses by the anti-malware tool may include killing any associated processes and deleting the files, killing the processes and quarantining the files so that they are not able to execute but are not deleted, or simply leaving whatever has been detected alone. Leaving the files intact is not a typical response but may be required as anti-malware tools do sometimes detect security tools and other files that are not malware, which we may want to leave alone and ignore in the future.

We can find anti-malware tools deployed on mobile devices, individual systems, and a variety of servers but monitored at the enterprise level as a matter of course for large enterprise environments in order to protect these systems. We

may also find such tools installed on proxy servers in order to filter malware out of the incoming and outgoing traffic. This is very common in the case of proxies for e-mail, as many items of malware use e-mail as a method of propagation. In the case where malware is detected by such a tool, we may see the e-mail rejected entirely, or we may merely see the malware stripped out of the message body or the offending attachment removed.

Executable space protection

Executable space protection is a hardware- and software-based technology that can be implemented by operating systems in order to foil attacks that use the same techniques we commonly used in malware. In short, executable space protection prevents certain portions of the memory used by the operating system and applications from being used to execute code. This means classic attacks such as buffer overflows that depend on being able to execute their commands in hijacked portions of memory may be prevented from functioning at all. Many operating systems also use address space layout randomization (ASLR) [4] in order to shift the contents of the memory in use around so that tampering with it is even more difficult.

More advanced

A buffer overflow attack works by inputting more data than an application is expecting from a particular input—for example, by entering 1000 characters into a field that was only expecting 10. Depending on how the application was written, we may find that the extra 990 characters are written somewhere into memory, perhaps over memory locations used by other applications or the operating system. It is sometimes possible to execute commands by specifically crafting the excess data.

Executable space protection requires two components to function: a hardware component and a software component. Both of the main CPU chip manufacturers, Intel and AMD, support executable space protection, with Intel calling it the Execute Disable (XD) bit [5] and AMD calling it Enhanced Virus Protection [6].

The software implementation of executable space prevention can be found in many common operating systems. Both executable space prevention and ASLR can be found in many operating systems from Microsoft and Apple, as well as a number of Linux distributions, just to name a few.

Software firewalls and host intrusion detection

In addition to the tools we can use on the network to detect and filter out undesirable traffic, such as firewalls and IDS, we can add another layer of security at the

host level by implementing a very similar set of tools here. Although we may often find firewalls and IDS implemented at the network level in the form of purpose-built appliances, the actual functions they perform are generally carried out via specialized software resident on the devices. Similar software can be installed directly onto the hosts residing on our networks.

Software firewalls

Properly configured software firewalls are a very useful additional layer of security we can add to the hosts residing on our networks. Such firewalls generally contain a subset of the features we might find on a large firewall appliance but are often capable of very similar packet filtering and stateful packet inspection. We often find the rulesets of such applications expressed in terms of the particular applications and ports allowed to send and receive traffic on the various network interfaces that exist on the host. Such softwares can range from the relatively simple versions that are built into and ship with common operating systems, such as Windows and OS X, to large versions intended for use on corporate networks that include centralized monitoring and the capability for considerably more complex rules and management options.

Host intrusion detection

HIDS are used to analyze the activities on or directed at the network interface of a particular host. They have many of the same advantages as network-based intrusion detection systems (NIDS) have but with a considerably reduced scope of operation. As with software firewalls, such tools may range from simple consumer versions to much more complex commercial versions that allow for centralized monitoring and management.

A potential flaw with centrally managed HIDS is that, in order for the software to report an attack to the management mechanism in real time, the information needs to be communicated over the network. If the host in question is being actively attacked via the same network we would report over, we may not be able to do this. We can attempt to mitigate such issues by sending a regular beacon from the device to the management mechanism, allowing us to assume a problem if we stop seeing multiple devices unexpectedly, but this might not be a complete approach.

Operating system security tools

As we discussed in our coverage of the tools we might use to evaluate our network security in Chapter 10, a number of the same or similar tools can also be used to assess the security of our hosts. We can use scanners to examine how our hosts interact with the rest of the devices on the network, vulnerability assessment

tools to help point out particular areas where we might find applications or services that may be open to attack, privilege escalation tools to gain unauthorized access on our systems, and various exploit frameworks to allow us access to a broad array of tools and attacks that might be used by those who would attempt to subvert our security. The tools we will discuss in this section do not resemble an exhaustive list, but we will hit a few of the highlights.

Scanners

We can use a large number of scanning tools to assist in detecting various security flaws when we are looking at hosts. Although we discussed this in Chapter 10 from a network perspective, such tools can also be used to enhance the security of our hosts. We can look for open ports and versions of services that are running, examine banners displayed by services for information, examine the information our systems display over the network, and perform a large number of similar tasks.

Earlier in this chapter, when we were discussing hardening, we looked at a very simple example of using Nmap to look at a device over the network in order to discover the ports that had services listening on them. Nmap actually has a very large and broad set of functionality and can give us considerably more information if we ask it to do so. In Figure 11.3, we can see the results of an Nmap

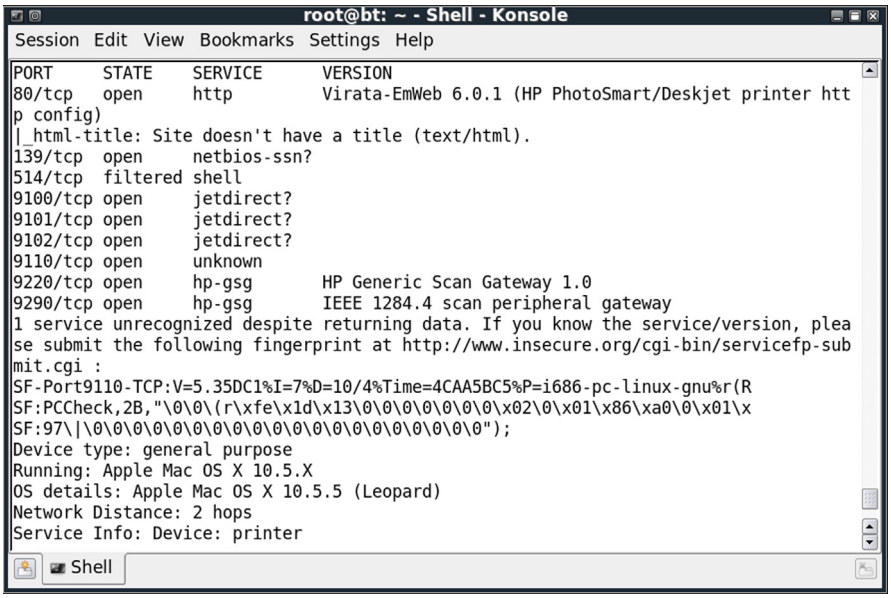

FIGURE 11.3

Nmap scan result.

scan directed against a network printer. In this case, we asked Nmap to also look for the particular versions of the services it found and to attempt to identify the operating system running on the device. If we look at port 9220 in the listing, we can see that the service is hp-gsg, which, although a bit cryptic, might give us somewhat of a clue that it is a service specific to HP printers, but if we look at the version information on the same line, we can see very specifically that the service is HP Generic Scan Gateway 1.0. Based on this information, we might have a much better chance of successfully being able to attack the device.

Alert!

Looking closer at the Nmap results in Figure 11.3, you will note that Nmap told us the device being scanned was a printer, but it also told us it was running Mac OS X as an operating system. Sometimes Nmap's OS fingerprints can be a little skewed from what is actually on the device, so it is often best to verify the output from Nmap with another tool if something looks odd.

In addition to the many features built into Nmap, we can create custom Nmap functionality of our own, through the use of the Nmap Scripting Engine (NSE).

Vulnerability assessment tools

Vulnerability assessment tools, which often include some portion of the feature set we might find in a tool such as Nmap, are aimed specifically at the task of finding and reporting network services on hosts that have known vulnerabilities.

One such well-known scanning tool is Tenable's Nessus. Although Nessus was, at one time, a free tool, it is no longer entirely so. Nessus is now primarily a commercial tool, with a limited free license available for noncommercial use. Nessus is chiefly a graphically interfaced vulnerability assessment tool, as shown in Figure 11.4. In essence, Nessus will conduct a port scan on a target, then attempt to determine what services and versions of service are running on any ports it finds open. Nessus will then report back with a specific list of vulnerabilities that we might find on a given device.

As we mentioned, Nessus, as a part of its feature set, includes a port scanner, as a port scan is needed in order to find the listening services before we can identify the vulnerabilities that might be resident in them. Nessus also includes some other functionalities, including the ability to add custom features to the tool through the Nessus Attack Scripting Language (NASL).

Exploit frameworks

A category of tools, or more accurately, a category of sets of tools, called an exploit framework, enjoyed a rise in popularity in the first few years of the 2000s and is still going strong. Many exploit frameworks provide a variety of tools,

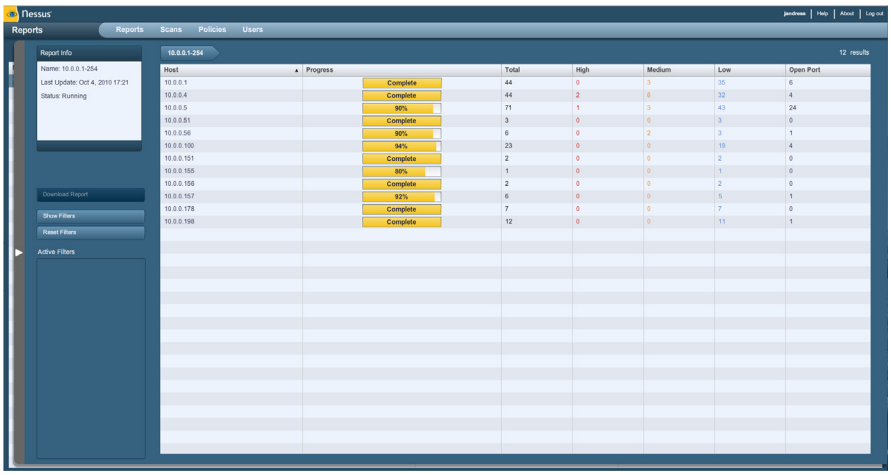

FIGURE 11.4

Nessus.

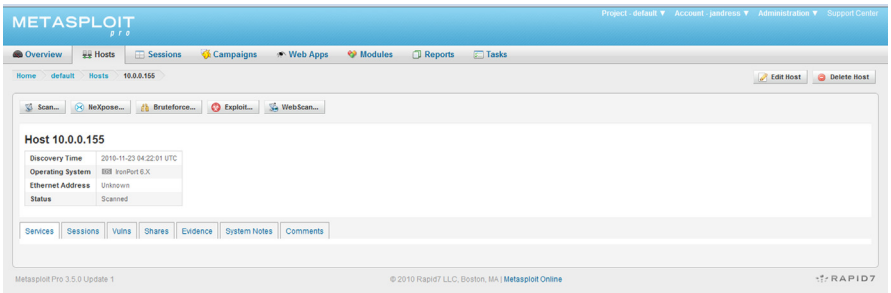

FIGURE 11.5

Metasploit Pro.

including network mapping tools, sniffers, and many more, but one of the main tools we can find in exploit frameworks is, logically, the exploit.

Exploits are small bits of software that take advantage of, or exploit, flaws in other software or applications in order to cause them to behave in ways that were not intended by their creators. Exploits are commonly used by attackers to gain access to systems or gain additional privileges on them when they already have access.

Exploit frameworks, such as Rapid7's Metasploit, as shown in Figure 11.5, Immunity CANVAS, and Core Impact provide large sets of prepackaged exploits in order to make them simple to use and to make a larger library available to us than we might have if we had to put them together individually. Many exploit frameworks come in the form of graphically interfaced tools that can be run in

much the same way that any other application functions. Some tools can even be configured to automatically seek out and attack systems, spreading further into the network as they gain additional access. We commonly see the use of exploit frameworks in penetration testing.

Operating system security in the real world

The operating system security measures we discussed in this chapter are in common use in companies around the globe. The various steps we went over when we discussed hardening operating systems are usually implemented by any competent organization that is building servers for deployment, particularly in cases where these servers will be Internet facing. Depending on the organization in question and its security posture, we may or may not find such measures to have been carried out on client machines. Although such basic hardening measures are a way in which we can increase our security with relative ease, we do so at the potential expense of ease of use and productivity.

The use of anti-malware tools, HIDS, and software firewalls is also rather ubiquitous in many organizations of any appreciable size. We will commonly see anti-malware tools installed on proxy servers filtering Web and mail traffic as it enters from the Internet. Without such tools in place, even if we have very strong border security in the form of firewalls and IDS, when something does manage to make it through these measures, it will cause great havoc on our internal networks.

The tools we discussed in this chapter and in Chapter 10 are some of the staples of the security industry. A huge number and variety of such tools might be used in any given environment for any number of uses, but taking the time to learn some of those that are more commonly seen, such as Nmap and Nessus, will be helpful to anyone entering the security field. We may see larger and costlier commercial tools at use in a given environment, but they will often be in use side by side with the old standbys.

SUMMARY

One of the primary tools we can use in our efforts to secure the operating systems for which we are responsible is hardening. The main tasks, when we seek to harden an operating system, are to remove all unnecessary software, remove all unessential services, alter the default accounts on the system, utilize the principle of least privilege, apply updates to software in an appropriate manner, and conduct logging and auditing.

We can also apply various additional layers of security to our operating systems in the form of additional software. We can install anti-malware tools in an

effort to detect, prevent, and remove malware when we encounter it. We can put firewall technology to use directly on our hosts, in order to filter out undesirable traffic as it enters or exits our network interfaces. We can also install HIDS in order to detect attacks as they come at us over the network.

In our efforts to secure our operating systems, we can make use of a variety of security tools in order to find the security flaws that might be present. A number of scanning tools are available, with Nmap being one of the most well known among them. We can also make use of vulnerability assessment tools in order to locate specific security flaws in our services or network-enabled software, such as Nessus. Additionally, we can use exploit frameworks to attack systems in an effort to gain access to them or to gain elevated privilege levels, with Metasploit being one of the better-known tools.

EXERCISES

1. What is a vector for malware propagation?
2. What is an exploit framework?
3. What is the difference between a port scanner and a vulnerability assessment tool?
4. Explain the concept of an attack surface.
5. Why might we want a firewall on our host if one already exists on the network?
6. What is operating system hardening?
7. What is the XD bit and why do we use it?
8. What does executable space protection do for us?
9. How does the principle of least privilege apply to operating system hardening?
10. Download Nmap from www.nmap.org and install it. Conduct a basic scan of scanme.nmap.org using either the Zenmap GUI or the command line. What ports can you find open?

References

[1] Schneider FB, editor. Trust in cyberspace. Washington, DC: National Academy Press; 1998, ISBN-13: 9780309065580.
[2] Falliere N, Murchu LO, Chien E. W32.Stuxnet Dossier. Symantec 2011.

[3] Barnes Ed. Mystery surrounds cyber missile that crippled Iran's nuclear weapons ambitions, Fox News, <www.foxnews.com/scitech/2010/11/26/secret-agent-crippled-irans-nuclear-ambitions/>; November 26, 2010 [accessed 24.10.13].
[4] Barrantes EG, Ackley DH, Palmer TS, Zovi DD, Forrest S, Stefanovic D. Randomized instruction setemulation to disrupt binary code injection attacks. Proceedings of the tenth ACM conference on computer and communications security; 2003. ISBN: 1581137389.
[5] Intel Corporation. Execute disable bit and enterprise security, Intel.com, <www.intel.com/technology/xdbit/index.htm>; 2011 [accessed 24.10.13].
[6] Advanced Micro Devices, Inc. Enhanced virus protection, AMD.com, <www.amd.com/us/products/technologies/enhanced-virus-protection/Pages/enhanced-virus-protection.aspx>; 2011 [accessed 24.10.13].

This page intentionally left blank