CHAPTER 12

OPERATING SYSTEM SECURITY

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- List the steps needed in the process of securing a system.
- Detail the need for planning system security.
- List the basic steps used to secure the base operating system.
- List the additional steps needed to secure key applications.
- List steps needed to maintain security.
- List some specific aspects of securing Unix/Linux systems.
- List some specific aspects of securing Windows systems.
- List steps needed to maintain security in virtualized systems.

Computer client and server systems are central components of the IT infrastructure for most organizations. The client systems provide access to organizational data and applications, supported by the servers housing those data and applications. However, given that most large software systems will almost certainly have a number of security weaknesses, as we discussed in Chapter 6 and in the previous two chapters, it is currently necessary to manage the installation and continuing operation of these systems to provide appropriate levels of security despite the expected presence of these vulnerabilities. In some circumstances we may be able to use systems designed and evaluated to provide security by design. We examine some of these possibilities in the next chapter.

In this chapter we discuss how to provide systems security as a **hardening** process that includes planning, installation, configuration, update, and maintenance of the operating system and the key applications in use, following the general approach detailed in [SCAR08]. We consider this process for the operating system, and then key applications in general, and then discuss some specific aspects in relation to Linux and Windows systems in particular. We conclude with a discussion on securing virtualized systems, where multiple virtual machines may execute on the one physical system.

We view a system as having a number of layers, with the physical hardware at the bottom; the base operating system above including privileged kernel code, APIs, and services; and finally user applications and utilities in the top layer, as shown in Figure 12.1. This figure also shows the presence of BIOS and possibly other code that is external to, and largely not visible from, the operating system

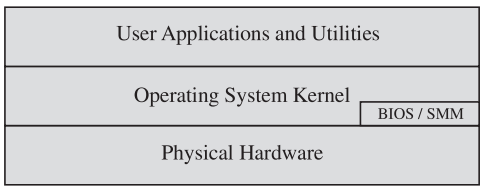

Figure 12.1 Operating System Security Layers

kernel, but which is used when booting the system or to support low-level hardware control. Each of these layers of code needs appropriate **hardening** measures in place to provide appropriate security services. And each layer is vulnerable to attack from below, should the lower layers not also be secured appropriately.

A number of reports note that the use of a small number of basic hardening measures can prevent a large proportion of the attacks seen in recent years. Since 2010 the Australian Signals Directorate (ASD) list of the "Top 35 Mitigation Strategies" has noted that implementing just the top four of these strategies would have prevented at least 85% of the targeted cyber intrusions investigated by ASD. Hence, since 2013 these top four strategies are mandatory for all Australian government agencies. These top four strategies are:

1. White-list approved applications.
2. Patch third-party applications and operating system vulnerabilities.
3. Restrict administrative privileges.
4. Create a defense-in-depth system.

We discuss all four of these strategies, and many others in the ASD list, in this chapter. Note that these strategies largely align with those in the "20 Critical Controls" developed by DHS, NSA, the Department of Energy, SANS, and others in the United States.

12.1 INTRODUCTION TO OPERATING SYSTEM SECURITY

As we noted above, computer client and server systems are central components of the IT infrastructure for most organizations, may hold critical data and applications, and are a necessary tool for the function of an organization. Accordingly, we need to be aware of the expected presence of vulnerabilities in operating systems and applications as distributed, and the existence of worms scanning for such vulnerabilities at high rates, such as we discussed in Section 6.3. Thus, it is quite possible for a system to be compromised during the installation process, before it can install the latest patches or implement other hardening measures. Hence, building and deploying a system should be a planned process designed to counter such a threat, and to maintain security during its operational lifetime.

[SCAR08] states that this process must:

- Assess risks and plan the system deployment.
- Secure the underlying operating system and then the key applications.
- Ensure any critical content is secured.
- Ensure appropriate network protection mechanisms are used.
- Ensure appropriate processes are used to maintain security.

While we address the selection of network protection mechanisms in Chapter 9, we examine the other items in the rest of this chapter.

12.2 SYSTEM SECURITY PLANNING

The first step in deploying new systems is planning. Careful planning will help ensure that the new system is as secure as possible, and complies with any necessary policies. This planning should be informed by a wider security assessment of the organization, since every organization has distinct security requirements and concerns. We discuss this wider planning process in Chapters 14 and 15.

The aim of the specific system installation planning process is to maximize security while minimizing costs. Wide experience shows that it is much more difficult and expensive to "retro-fit" security at a later time, than it is to plan and provide it during the initial deployment process. This planning process needs to determine the security requirements for the system, its applications and data, and of its users. This then guides the selection of appropriate software for the operating system and applications, and provides guidance on appropriate user configuration and access control settings. It also guides the selection of other **hardening** measures required. The plan also needs to identify appropriate personnel to install and manage the system, noting the skills required and any training needed.

[SCAR08] provides a list of items that should be considered during the system security planning process. While its focus is on secure server deployment, much of the list applies equally well to client system design. This list includes consideration of:

- The purpose of the system, the type of information stored, the applications and services provided, and their security requirements.
- The categories of users of the system, the privileges they have, and the types of information they can access.
- How the users are authenticated.
- How access to the information stored on the system is managed.
- What access the system has to information stored on other hosts, such as file or database servers, and how this is managed.
- Who will administer the system, and how they will manage the system (via local or remote access).
- Any additional security measures required on the system, including the use of host firewalls, anti-virus or other malware protection mechanisms, and logging.

12.3 OPERATING SYSTEMS HARDENING

The first critical step in securing a system is to secure the base operating system upon which all other applications and services rely. A good security foundation needs a properly installed, patched, and configured operating system. Unfortunately, the default configuration for many operating systems often maximizes ease of use and functionality, rather than security. Further, since every organization has its own security needs, the appropriate security profile, and hence configuration, will also differ. What is required for a particular system should be identified during the planning phase, as we have just discussed.

While the details of how to secure each specific operating system differ, the broad approach is similar. Appropriate security configuration guides and checklists exist for most common operating systems, and these should be consulted, though always informed by the specific needs of each organization and their systems. In some cases, automated tools may be available to further assist in securing the system configuration.

[SCAR08] suggests the following basic steps that should be used to secure an operating system:

- Install and patch the operating system.
- Harden and configure the operating system to adequately address the identified security needs of the system by:
  - Removing unnecessary services, applications, and protocols.
  - Configuring users, groups, and permissions.
  - Configuring resource controls.
- Install and configure additional security controls, such as anti-virus, host-based firewalls, and intrusion detection systems (IDS), if needed.
- Test the security of the basic operating system to ensure that the steps taken adequately address its security needs.

Operating System Installation: Initial Setup and Patching

System security begins with the installation of the operating system. As we have already noted, a network connected, unpatched system, is vulnerable to exploit during its installation or continued use. Hence it is important that the system not be exposed while in this vulnerable state. Ideally new systems should be constructed on a protected network. This may be a completely isolated network, with the operating system image and all available patches transferred to it using removable media such as DVDs or USB drives. Given the existence of malware that can propagate using removable media, as we discuss in Chapter 6, care is needed to ensure the media used here is not so infected. Alternatively, a network with severely restricted access to the wider Internet may be used. Ideally it should have no inbound access, and have outbound access only to the key sites needed for the system installation and patching process. In either case, the full installation and **hardening** process should occur before the system is deployed to its intended, more accessible, and hence vulnerable, location.

The initial installation should install the minimum necessary for the desired system, with additional software packages included only if they are required for the function of the system. We explore the rationale for minimizing the number of packages on the system shortly.

The overall boot process must also be secured. This may require adjusting options on, or specifying a password required for changes to, the BIOS code used when the system initially boots. It may also require limiting which media the system is normally permitted to boot from. This is necessary to prevent an attacker from changing the boot process to install a covert hypervisor, such as we discussed in Section 6.8, or to just boot a system of their choice from external media in order to bypass the normal system access controls on locally stored data. The

use of a cryptographic file system may also be used to address this threat, as we note later.

Care is also required with the selection and installation of any additional device driver code, since this executes with full kernel level privileges, but is often supplied by a third party. The integrity and source of such driver code must be carefully validated given the high level of trust it has. A malicious driver can potentially bypass many security controls to install malware. This was done in both the Blue Pill demonstration rootkit, which we discussed in Section 6.8, and the Stuxnet worm, which we described in Section 6.3.

Given the continuing discovery of software and other vulnerabilities for commonly used operating systems and applications, it is critical that the system be kept as up to date as possible, with all critical security related **patches** installed. Indeed, doing this addresses one of the top four key ASD mitigation strategies we listed previously. Nearly all commonly used systems now provide utilities that can automatically download and install security updates. These tools should be configured and used to minimize the time any system is vulnerable to weaknesses for which patches are available.

Note that on change-controlled systems, you should not run automatic updates, because security patches can, on rare but significant occasions, introduce instability. For systems on which availability and uptime are of paramount importance, therefore, you should stage and validate all patches on test systems before deploying them in production.

Remove Unnecessary Services, Application, and Protocols

Because any of the software packages running on a system may contain software vulnerabilities, clearly if fewer software packages are available to run, then the risk is reduced. There is clearly a balance between usability, providing all software that may be required at some time, with security, and a desire to limit the amount of software installed. The range of services, applications, and protocols required will vary widely between organizations, and indeed between systems within an organization. The system planning process should identify what is actually required for a given system, so that a suitable level of functionality is provided, while eliminating software that is not required to improve security.

The default configuration for most distributed systems is set to maximize ease of use and functionality, rather than security. When performing the initial installation, the supplied defaults should not be used, but rather the installation should be customized so that only the required packages are installed. If additional packages are needed later, they can be installed when they are required. [SCAR08] and many of the security hardening guides provide lists of services, applications, and protocols that should not be installed if not required.

[SCAR08] also states a strong preference for not installing unwanted software, rather than installing and then later removing or disabling it. It argues this preference because they note that many uninstall scripts fail to completely remove all components of a package. They also note that disabling a service means that while it is not available as an initial point of attack, should an attacker succeed in gaining some access to a system, then disabled software could be re-enabled and used to

further compromise a system. It is better for security if unwanted software is not installed, and thus not available for use at all.

Configure Users, Groups, and Authentication

Not all users with access to a system will have the same access to all data and resources on that system. All modern operating systems implement **access controls** to data and resources, as we discuss in Chapter 4. Nearly all provide some form of discretionary access controls. Some systems may provide role-based or mandatory access control mechanisms as well.

The system planning process should consider the categories of users on the system, the privileges they have, the types of information they can access, and how and where they are defined and authenticated. Some users will have elevated privileges to administer the system; others will be normal users, sharing appropriate access to files and other data as required; and there may even be guest accounts with very limited access. The third of the four key ASD mitigation strategies is to restrict elevated privileges to only those users that require them. Further, it is highly desirable that such users only access elevated privileges when needed to perform some task that requires them, and to otherwise access the system as a normal user. This improves security by providing a smaller window of opportunity for an attacker to exploit the actions of such privileged users. Some operating systems provide special tools or access mechanisms to assist administrative users to elevate their privileges only when necessary, and to appropriately log these actions.

One key decision is whether the users, the groups they belong to, and their authentication methods are specified locally on the system or will use a centralized authentication server. Whichever is chosen, the appropriate details are now configured on the system.

Also at this stage, any default accounts included as part of the system installation should be secured. Those which are not required should be either removed or at least disabled. System accounts that manage services on the system should be set so they cannot be used for interactive logins. And any passwords installed by default should be changed to new values with appropriate security.

Any policy that applies to authentication credentials, and especially to password security, is also configured. This includes details of which authentication methods are accepted for different methods of account access. And it includes details of the required length, complexity, and age allowed for passwords. We discuss some of these issues in Chapter 3.

Configure Resource Controls

Once the users and their associated groups are defined, appropriate **permissions** can be set on data and resources to match the specified policy. This may be to limit which users can execute some programs, especially those that modify the system state. Or it may be to limit which users can read or write data in certain directory trees. Many of the security hardening guides provide lists of recommended changes to the default access configuration to improve security.

Install Additional Security Controls

Further security improvement may be possible by installing and configuring additional security tools such as anti-virus software, host-based firewalls, IDS or IPS software, or application white-listing. Some of these may be supplied as part of the operating systems installation, but not configured and enabled by default. Others are third-party products that are acquired and used.

Given the widespread prevalence of malware, as we discuss in Chapter 6, appropriate anti-virus (which as noted addresses a wide range of malware types) is a critical security component on many systems. Anti-virus products have traditionally been used on Windows systems, since their high use made them a preferred target for attackers. However, the growth in other platforms, particularly smartphones, has led to more malware being developed for them. Hence appropriate anti-virus products should be considered for any system as part of its security profile.

Host-based firewalls, IDS, and IPS software also may improve security by limiting remote network access to services on the system. If remote access to a service is not required, though some local access is, then such restrictions help secure such services from remote exploit by an attacker. Firewalls are traditionally configured to limit access by port or protocol, from some or all external systems. Some may also be configured to allow access from or to specific programs on the systems, to further restrict the points of attack, and to prevent an attacker installing and accessing their own malware. IDS and IPS software may include additional mechanisms such as traffic monitoring, or file integrity checking to identify and even respond to some types of attack.

Another additional control is to white-list applications. This limits the programs that can execute on the system to just those in an explicit list. Such a tool can prevent an attacker installing and running their own malware, and is the first of the four key ASD mitigation strategies. While this will improve security, it functions best in an environment with a predictable set of applications that users require. Any change in software usage would require a change in the configuration, which may result in increased IT support demands. Not all organizations or all systems will be sufficiently predictable to suit this type of control.

Test the System Security

The final step in the process of initially securing the base operating system is security testing. The goal is to ensure that the previous security configuration steps are correctly implemented, and to identify any possible vulnerabilities that must be corrected or managed.

Suitable checklists are included in many security hardening guides. There are also programs specifically designed to review a system to ensure that a system meets the basic security requirements, and to scan for known vulnerabilities and poor configuration practices. This should be done following the initial hardening of the system, and then repeated periodically as part of the security maintenance process.

12.4 APPLICATION SECURITY

Once the base operating system is installed and appropriately secured, the required services and applications must next be installed and configured. The steps for this very much mirror the list already given in the previous section. The concern, as with the base operating system, is to only install software on the system that is required to meet its desired functionality, in order to reduce the number of places vulnerabilities may be found. Software that provides remote access or service is of particular concern, since an attacker may be able to exploit this to gain remote access to the system. Hence any such software needs to be carefully selected and configured, and updated to the most recent version available.

Each selected service or application must be installed, and then patched to the most recent supported secure version appropriate for the system. This may be from additional packages provided with the operating system distribution, or from a separate third-party package. As with the base operating system, utilizing an isolated, secure build network is preferred.

Application Configuration

Any application specific configuration is then performed. This may include creating and specifying appropriate data storage areas for the application, and making appropriate changes to the application or service default configuration details.

Some applications or services may include default data, scripts, or user accounts. These should be reviewed, and only retained if required, and suitably secured. A well-known example of this is found with Web servers, which often include a number of example scripts, quite a few of which are known to be insecure. These should not be used as supplied, but should be removed unless needed and secured.

As part of the configuration process, careful consideration should be given to the access rights granted to the application. Again, this is of particular concern with remotely accessed services, such as Web and file transfer services. The server application should not be granted the right to modify files, unless that function is specifically required. A very common configuration fault seen with Web and file transfer servers is for all the files supplied by the service to be owned by the same "user" account that the server executes as. The consequence is that any attacker able to exploit some vulnerability in either the server software or a script executed by the server may be able to modify any of these files. The large number of "Web defacement" attacks is clear evidence of this type of insecure configuration. Much of the risk from this form of attack is reduced by ensuring that most of the files can only be read, but not written, by the server. Only those files that need to be modified, to store uploaded form data for example, or logging details, should be writeable by the server. Instead the files should mostly be owned and modified by the users on the system who are responsible for maintaining the information.

Encryption Technology

Encryption is a key enabling technology that may be used to secure data both in transit and when stored, as we discuss in Chapter 2 and in Parts Four and Five. If such technologies are required for the system, then they must be configured, and appropriate cryptographic keys created, signed, and secured.

If secure network services are provided, most likely using either TLS or IPsec, then suitable public and private keys must be generated for each of them. Then X.509 certificates are created and signed by a suitable certificate authority, linking each service identity with the public key in use, as we discuss in Section 23.2. If secure remote access is provided using Secure Shell (SSH), then appropriate server, and possibly client keys, must be created.

Cryptographic file systems are another use of encryption. If desired, then these must be created and secured with suitable keys.

12.5 SECURITY MAINTENANCE

Once the system is appropriately built, secured, and deployed, the process of maintaining security is continuous. This results from the constantly changing environment, the discovery of new vulnerabilities, and hence exposure to new threats. [SCAR08] suggests that this process of security maintenance includes the following additional steps:

- Monitoring and analyzing logging information
- Performing regular backups
- Recovering from security compromises
- Regularly testing system security
- Using appropriate software maintenance processes to patch and update all critical software, and to monitor and revise configuration as needed

We have already noted the need to configure automatic **patching** and update where possible, or to have a process to manually test and install patches on configuration controlled systems, and that the system should be regularly tested using checklist or automated tools where possible. We discuss the process of incident response in Section 15.5. We now consider the critical logging and backup procedures.

Logging

[SCAR08] notes that "logging is a cornerstone of a sound security posture." **Logging** is a reactive control that can only inform you about bad things that have already happened. But effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and thus most effectively focus their remediation and recovery efforts. The key is to ensure you capture the correct data in the logs, and are then able to appropriately monitor and analyze this data. Logging information can be generated by the system, network, and applications. The range of logging data acquired should be determined during the system planning stage, as it depends on the security requirements and information sensitivity of the server.

Logging can generate significant volumes of information. It is important that sufficient space is allocated for them. A suitable automatic log rotation and archive system should also be configured to assist in managing the overall size of the logging information.

Manual analysis of logs is tedious and is not a reliable means of detecting adverse events. Rather, some form of automated analysis is preferred, as it is more likely to identify abnormal activity.

We discuss the process of logging further in Chapter 18.

Data Backup and Archive

Performing regular backups of data on a system is another critical control that assists with maintaining the integrity of the system and user data. There are many reasons why data can be lost from a system, including hardware or software failures, or accidental or deliberate corruption. There may also be legal or operational requirements for the retention of data. **Backup** is the process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks. **Archive** is the process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data. These processes are often linked and managed together, although they do address distinct needs.

The needs and policy relating to backup and archive should be determined during the system planning stage. Key decisions include whether the backup copies are kept online or offline, and whether copies are stored locally or transported to a remote site. The trade-offs include ease of implementation and cost versus greater security and robustness against different threats.

A good example of the consequences of poor choices here was seen in the attack on an Australian hosting provider in early 2011. The attackers destroyed not only the live copies of thousands of customer's sites, but also all of the online backup copies. As a result, many customers who had not kept their own backup copies lost all of their site content and data, with serious consequences for many of them, and for the hosting provider as well. In other examples, many organizations that only retained onsite backups have lost all their data as a result of fire or flooding in their IT center. These risks must be appropriately evaluated.

12.6 LINUX/UNIX SECURITY

Having discussed the process of enhancing security in operating systems through careful installation, configuration, and management, we now consider some specific aspects of this process as it relates to Unix and Linux systems. Beyond the general guidance in this section, we provide a more detailed discussion of Linux security mechanisms in Chapter 25.

There are a large range of resources available to assist **administrators** of these systems, including many texts, for example [NEME10], online resources such as the "Linux Documentation Project," and specific system hardening guides such as those provided by the "NSA—Security Configuration Guides." These resources should be used as part of the system security planning process in order to incorporate procedures appropriate to the security requirements identified for the system.

Patch Management

Ensuring that system and application code is kept up to date with security patches is a widely recognized and critical control for maintaining security.

Modern Unix and Linux distributions typically include tools for automatically downloading and installing software updates, including security updates, which can minimize the time a system is vulnerable to known vulnerabilities for which patches exist. For example, Red Hat, Fedora, and CentOS include `up2date` or `yum`; SuSE includes `yast`; and Debian uses `apt-get`, though you must run it as a cron job for automatic updates. It is important to configure whichever update tool is provided on the distribution in use, to install at least critical security patches in a timely manner.

As noted earlier, change-controlled systems should not run automatic updates, because they may possibly introduce instability. Such systems should validate all patches on test systems before deploying them to production systems.

Application and Service Configuration

Configuration of applications and services on Unix and Linux systems is most commonly implemented using separate text files for each application and service. System-wide configuration details are generally located either in the /etc directory or in the installation tree for a specific application. Where appropriate, individual user configurations that can override the system defaults are located in hidden "dot" files in each user's home directory. The name, format, and usage of these files are very much dependent on the particular system version and applications in use. Hence the systems administrators responsible for the secure configuration of such a system must be suitably trained and familiar with them.

Traditionally, these files were individually edited using a text editor, with any changes made taking effect either when the system was next rebooted or when the relevant process was sent a signal indicating that it should reload its configuration settings. Current systems often provide a GUI interface to these configuration files to ease management for novice administrators. Using such a manager may be appropriate for small sites with a limited number of systems. Organizations with larger numbers of systems may instead employ some form of centralized management, with a central repository of critical configuration files that can be automatically customized and distributed to the systems they manage.

The most important changes needed to improve system security are to disable services, especially remotely accessible services, and applications, that are not required, and to then ensure that applications and services that are needed are appropriately configured, following the relevant security guidance for each. We provide further details on this in Section 25.5.

Users, Groups, and Permissions

As we describe in Sections 4.5 and 25.3, Unix and Linux systems implement discretionary access control to all file system resources. These include not only files and directories but devices, processes, memory, and indeed most system resources. Access is specified as granting read, write, and execute **permissions** to each of

owner, group, and others, for each resource, as shown in Figure 4.6. These are set using the `chmod` command. Some systems also support extended file attributes with access control lists that provide more flexibility, by specifying these permissions for each entry in a list of users and groups. These extended access rights are typically set and displayed using the `getfacl` and `setfacl` commands. These commands can also be used to specify set user or set group permissions on the resource.

Information on user accounts and group membership are traditionally stored in the `/etc/passwd` and `/etc/group` files, though modern systems also have the ability to import these details from external repositories queried using LDAP or NIS for example. These sources of information, and indeed of any associated authentication credentials, are specified in the PAM (pluggable authentication module) configuration for the system, often using text files in the `/etc/pam.d` directory.

In order to partition access to information and resources on the system, users need to be assigned to appropriate groups granting them any required access. The number and assignments to groups should be decided during the system security planning process, and then configured in the appropriate information repository, whether locally using the configuration files in `/etc`, or on some centralized database. At this time, any default or generic users supplied with the system should be checked, and removed if not required. Other accounts that are required, but are not associated with a user that needs to login, should have login capability disabled, and any associated password or authentication credential removed.

Guides to hardening Unix and Linux systems also often recommend changing the access permissions for critical directories and files, in order to further limit access to them. Programs that set user (setuid) to root or set group (setgid) to a privileged group are key target for attackers. As we detail in Sections 4.5 and 25.3, such programs execute with superuser rights, or with access to resources belonging to the privileged group, no matter which user executes them. A software vulnerability in such a program can potentially be exploited by an attacker to gain these elevated privileges. This is known as a local exploit. A software vulnerability in a network server could be triggered by a remote attacker. This is known as a remote exploit.

It is widely accepted that the number and size of setuid root programs in particular should be minimized. They cannot be eliminated, as superuser privileges are required to access some resources on the system. The programs that manage user login, and allow network services to bind to privileged ports, are examples. However, other programs, that were once setuid root for programmer convenience, can function as well if made setgid to a suitable privileged group that has the necessary access to some resource. Programs to display system state, or deliver mail, have been modified in this way. System hardening guides may recommend further changes and indeed the removal of some such programs that are not required on a particular system.

Remote Access Controls

Given that remote exploits are of concern, it is important to limit access to only those services required. This function may be provided by a perimeter firewall, as we discussed in Chapter 9. However, host-based firewall or network access control mechanisms may provide additional defences. Unix and Linux systems support several alternatives for this.

The TCP Wrappers library and `tcpd` daemon provide one mechanism that network servers may use. Lightly loaded services may be "wrapped" using `tcpd`, which listens for connection requests on their behalf. It checks that any request is permitted by configured policy before accepting it and invoking the server program to handle it. Requests that are rejected are logged. More complex and heavily loaded servers incorporate this functionality into their own connection management code, using the TCP Wrappers library, and the same policy configuration files. These files are `/etc/hosts.allow` and `/etc/hosts.deny`, which should be set as policy requires.

There are several host firewall programs that may be used. Linux systems primarily use the `iptables` program to configure the `netfilter` kernel module. This provides comprehensive, though complex, stateful packet filtering, monitoring, and modification capabilities. BSD-based systems (including MacOSX) now use the `pf` program with similar capabilities. Most systems provide an administrative utility to generate common configurations and to select which services will be permitted to access the system. These should be used unless there are non-standard requirements, given the skill and knowledge needed to run these programs to edit their configuration files.

Logging and Log Rotation

Most applications can be configured to log with levels of detail ranging from "debugging" (maximum detail) to "none." Some middle setting is usually the best choice, but you should not assume that the default setting is necessarily appropriate.

In addition, many applications allow you to specify either a dedicated file to write application event data to, or a syslog facility to use when writing log data to `/dev/log` (see Section 25.5). If you wish to handle system logs in a consistent, centralized manner, it is usually preferable for applications to send their log data to `/dev/log`. Note, however, that `logrotate` (also discussed in Section 25.5) can be configured to rotate *any* logs on the system, whether written by `syslogd`, `Syslog-NG`, or individual applications.

Application Security Using a chroot jail

Some network accessible services do not require access to the full file-system, but rather only need a limited set of data files and directories for their operation. FTP is a common example of such a service. It provides the ability to download files from, and upload files to, a specified directory tree. If such a server were compromised and had access to the entire system, an attacker could potentially access and compromise data elsewhere. Unix and Linux systems provide a mechanism to run such services in a **chroot** jail, which restricts the server's view of the file system to just a specified portion. This is done using the **chroot** system call that confines a process to some subset of the file system by mapping the root of the filesystem `"/"` to some other directory (e.g., `/srv/ftp/public`). To the "chrooted" server, everything in this chroot jail appears to actually be in / (e.g., the "real" directory `/srv/ftp/public/etc/myconfigfile` appears as `/etc/myconfigfile` in the chroot jail). Files in directories outside the chroot jail (e.g., `/srv/www` or `/etc`.) are not visible or reachable at all.

Chrooting therefore helps contain the effects of a given server being compromised or hijacked. The main disadvantage of this method is added complexity: a

number of files (including all executable libraries used by the server), directories, and devices needed must be copied into the chroot jail. Determining just what needs to go into the jail for the server to work properly can be tricky, though detailed procedures for chrooting many different applications are available.

Troubleshooting a chrooted application can also be difficult. Even if an application explicitly supports this feature, it may behave in unexpected ways when run chrooted. Note also that if the chrooted process runs as root, it can "break out" of the chroot jail with little difficulty. Still, the advantages usually far outweigh the disadvantages of chrooting network services.

Security Testing

The system hardening guides such as those provided by the "NSA—Security Configuration Guides" include security checklists for a number of Unix and Linux distributions that may be followed.

There are also a number of commercial and open-source tools available to perform system security scanning and vulnerability testing. One of the best known is "Nessus." This was originally an open-source tool, which was commercialized in 2005, though some limited free-use versions are available. "Tripwire" is a well-known file integrity checking tool that maintains a database of cryptographic hashes of monitored files, and scans to detect any changes, whether as a result of malicious attack, or simply accidental or incorrectly managed update. This again was originally an open-source tool, which now has both commercial and free variants available. The "Nmap" network scanner is another well-known and deployed assessment tool that focuses on identifying and profiling hosts on the target network, and the network services they offer.

12.7 WINDOWS SECURITY

We now consider some specific issues with the secure installation, configuration, and management of Microsoft Windows systems. These systems have for many years formed a significant portion of all "general purpose" system installations. Hence, they have been specifically targeted by attackers, and consequently security countermeasures are needed to deal with these challenges. The process of providing appropriate levels of security still follows the general outline we describe in this chapter. Beyond the general guidance in this section, we provide more detailed discussion of Windows security mechanisms later in Chapter 26.

Again, there are a large range of resources available to assist **administrators** of these systems, including reports such as [SYMA07], online resources such as the "Microsoft Security Tools & Checklists," and specific system hardening guides such as those provided by the "NSA—Security Configuration Guides."

Patch Management

The "Windows Update" service and the "Windows Server Update Services" assist with the regular maintenance of Microsoft software, and should be configured and used. Many other third-party applications also provide automatic update support, and these should be enabled for selected applications.

Users Administration and Access Controls

Users and groups in Windows systems are defined with a Security ID (SID). This information may be stored and used locally, on a single system, in the Security Account Manager (SAM). It may also be centrally managed for a group of systems belonging to a domain, with the information supplied by a central Active Directory (AD) system using the LDAP protocol. Most organizations with multiple systems will manage them using domains. These systems can also enforce common policy on users on any system in the domain. We further explore the Windows security architecture in Section 26.1.

Windows systems implement discretionary access controls to system resources such as files, shared memory, and named pipes. The access control list has a number of entries that may grant or deny access rights to a specific SID, which may be for an individual user or for some group of users. Windows Vista and later systems also include mandatory integrity controls. These label all objects, such as processes and files, and all users, as being of low, medium, high, or system integrity level. Then whenever data is written to an object, the system first ensures that the subject's integrity is equal or higher than the object's level. This implements a form of the Biba Integrity model we discuss in Section 13.2 that specifically targets the issue of untrusted remote code executing in, for example Windows Internet Explorer, trying to modify local resources.

Windows systems also define privileges, which are system wide and granted to user accounts. Examples of privileges include the ability to backup the computer (which requires overriding the normal access controls to obtain a complete backup), or the ability to change the system time. Some privileges are considered dangerous, as an attacker may use them to damage the system. Hence they must be granted with care. Others are regarded as benign, and may be granted to many or all user accounts.

As with any system, hardening the system configuration can include further limiting the rights and privileges granted to users and groups on the system. As the access control list gives deny entries greater precedence, you can set an explicit deny permission to prevent unauthorized access to some resource, even if the user is a member of a group that otherwise grants access.

When accessing files on a shared resource, a combination of share and NTFS **permissions** may be used to provide additional security and granularity. For example, you can grant full control to a share, but read-only access to the files within it. If access-based enumeration is enabled on shared resources, it can automatically hide any objects that a user is not permitted to read. This is useful with shared folders containing many users' home directories, for example.

You should also ensure users with administrative rights only use them when required, and otherwise access the system as a normal user. The User Account Control (UAC) provided in Vista and later systems assists with this requirement. These systems also provide Low Privilege Service Accounts that may be used for long-lived service processes, such as file, print, and DNS services that do not require elevated privileges.

Application and Service Configuration

Unlike Unix and Linux systems, much of the configuration information in Windows systems is centralized in the Registry, which forms a database of keys and values that may be queried and interpreted by applications on these systems.

Changes to these values can be made within specific applications, setting preferences in the application that are then saved in the registry using the appropriate keys and values. This approach hides the detailed representation from the administrator. Alternatively, the registry keys can be directly modified using the "Registry Editor." This approach is more useful for making bulk changes, such as those recommended in hardening guides. These changes may also be recorded in a central repository, and pushed out whenever a user logs in to a system within a network domain.

Other Security Controls

Given the predominance of malware that targets Windows systems, it is essential that suitable anti-virus, anti-spyware, personal firewall, and other malware and attack detection and handling software packages are installed and configured on such systems. This is clearly needed for network connected systems, as shown by the high-incidence numbers in reports such as [SYMA13]. However, as the Stuxnet attacks in 2010 show, even isolated systems updated using removable media are vulnerable, and thus must also be protected.

Current generation Windows systems include some basic firewall and malware countermeasure capabilities, which should certainly be used at a minimum. However, many organizations find that these should be augmented with one or more of the many commercial products available. One issue of concern is undesirable interactions between anti-virus and other products from multiple vendors. Care is needed when planning and installing such products to identify possible adverse interactions, and to ensure the set of products in use are compatible with each other.

Windows systems also support a range of cryptographic functions that may be used where desirable. These include support for encrypting files and directories using the Encrypting File System (EFS), and for full-disk encryption with AES using BitLocker.

Security Testing

The system hardening guides such as those provided by the "NSA—Security Configuration Guides" also include security checklists for various versions of Windows.

There are also a number of commercial and open-source tools available to perform system security scanning and vulnerability testing of Windows systems. The "Microsoft Baseline Security Analyzer" is a simple, free, easy-to-use tool that aims to help small- to medium-sized businesses improve the security of their systems by checking for compliance with Microsoft's security recommendations. Larger organizations are likely better served using one of the larger, centralized, commercial security analysis suites available.

12.8 VIRTUALIZATION SECURITY

Virtualization refers to a technology that provides an abstraction of the computing resources used by some software, which thus runs in a simulated environment called a virtual machine (VM). There are many types of virtualization; however, in

this section we are most interested in full virtualization. This allows multiple full operating system instances to execute on virtual hardware, supported by a hypervisor that manages access to the actual physical hardware resources. Benefits arising from using virtualization include better efficiency in the use of the physical system resources than is typically seen using a single operating system instance. This is particularly evident in the provision of virtualized server systems. **Virtualization** can also provide support for multiple distinct operating systems and associated applications on the one physical system. This is more commonly seen on client systems.

There are a number of additional security concerns raised in virtualized systems, as a consequence both of the multiple operating systems executing side by side and of the presence of the virtualized environment and hypervisor as a layer below the operating system kernels and the security services they provide. [CLEE09] presents a survey of some of the security issues arising from such a use of virtualization, a number of which we will discuss further.

Virtualization Alternatives

There are many forms of creating a simulated, virtualized environment. These include **application virtualization**, as provided by the Java Virtual Machine environment. This allows applications written for one environment, to execute on some other operating system. It also includes **full virtualization**, in which multiple full operating system instances execute in parallel. Each of these guest operating systems, along with their own set of applications, executes in its own VM on virtual hardware. These **guest OSs** are managed by a **hypervisor**, or **virtual machine monitor** (VMM), that coordinates access between each of the guests and the actual physical hardware resources, such as CPU, memory, disk, network, and other attached devices. The hypervisor provides a similar hardware interface as that seen by operating systems directly executing on the actual hardware. As a consequence, little if any modification is needed to the guest OSs and their applications. Recent generations of CPUs provide special instructions that improve the efficiency of hypervisor operation.

Full virtualization systems may be further divided into native virtualization systems, in which the hypervisor executes directly on the underlying hardware, as we show in Figure 12.2, and hosted virtualization systems, in which the hypervisor executes as just another application on a host OS that is running on the underlying hardware, as we show in Figure 12.3. **Native virtualization** systems are typically seen in servers, with the goal of improving the execution efficiency of the hardware.

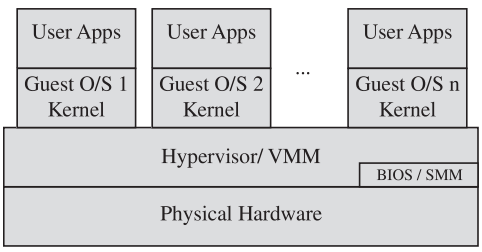

Figure 12.2 Native Virtualization Security Layers

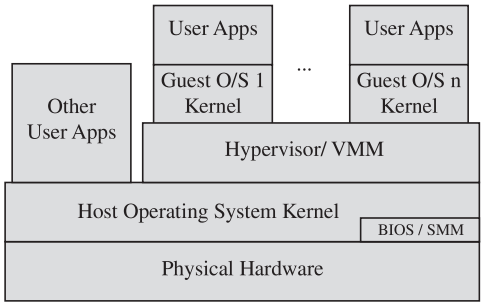

Figure 12.3 Hosted Virtualization Security Layers

They are arguably also more secure, as they have fewer additional layers than the alternative hosted approach. **Hosted virtualization** systems are more common in clients, where they run along side other applications on the host OS, and are used to support applications for alternate operating system versions or types. As this approach adds additional layers with the host OS under, and other host applications beside, the hypervisor, this may result in increased security concerns.

In virtualized systems, the available hardware resources must be appropriately shared between the various guest OSs. These include CPU, memory, disk, network, and other attached devices. CPU and memory are generally partitioned between these, and scheduled as required. Disk storage may be partitioned, with each guest having exclusive use of some disk resources. Alternatively, a "virtual disk" may be created for each guest, which appears to it as a physical disk with a full file-system, but is viewed externally as a single "disk image" file on the underlying file-system. Attached devices such as optical disks or USB devices are generally allocated to a single guest OS at a time. Several alternatives exist for providing network access. The guest OS may have direct access to distinct network interface cards on the system; the hypervisor may mediate access to shared interfaces; or the hypervisor may implement virtual network interface cards for each guest, routing traffic between guests as required. This last approach is quite common, and arguably the most efficient since traffic between guests does not need to be relayed via external network links. It does have security consequences in that this traffic is not subject to monitoring by probes attached to networks, such as we discussed in Chapter 9. Hence alternative, host-based probes would be needed in such a system if such monitoring is required.

Virtualization Security Issues

[CLEE09] and [SCAR11] both detail a number of security concerns that result from the use of virtualized systems, including:

- Guest OS isolation, ensuring that programs executing within a guest OS may only access and use the resources allocated to it, and not covertly interact with programs or data either in other guest OSs or in the hypervisor.
- Guest OS monitoring by the hypervisor, which has privileged access to the programs and data in each guest OS, and must be trusted as secure from subversion and compromised use of this access.

- Virtualized environment security, particularly image and snapshot management, which attackers may attempt to view or modify.

These security concerns may be regarded as an extension of the concerns we have already discussed with securing operating systems and applications. If a particular operating system and application configuration is vulnerable when running directly on hardware in some context, it will most likely also be vulnerable when running in a virtualized environment. And should that system actually be compromised, it would be at least as capable of attacking other nearby systems, whether they are also executing directly on hardware or running as other guests in a virtualized environment. The use of a virtualized environment may improve security by further isolating network traffic between guests than would be the case when such systems run natively, and from the ability of the hypervisor to transparently monitor activity on all guests OS. However, the presence of the virtualized environment and the hypervisor may reduce security if vulnerabilities exist within it which attackers may exploit. Such vulnerabilities could allow programs executing in a guest to covertly access the hypervisor, and hence other guest OS resources. This is known as VM escape, and is of concern, as we discussed in Section 6.8. Virtualized systems also often provide support for suspending an executing guest OS in a snapshot, saving that image, and then restarting execution at a later time, possibly even on another system. If an attacker can view or modify this image, they can compromise the security of the data and programs contained within it.

Thus the use of virtualization adds additional layers of concern, as we have previously noted. Securing virtualized systems means extending the security process to secure and harden these additional layers. In addition to securing each guest operating system and applications, the virtualized environment and the hypervisor must also be secured.

Securing Virtualization Systems

[SCAR11] provides guidance for providing appropriate security in virtualized systems, and states that organizations using virtualization should:

- Carefully plan the security of the virtualized system
- Secure all elements of a full virtualization solution, including the hypervisor, guest OSs, and virtualized infrastructure, and maintain their security
- Ensure that the hypervisor is properly secured
- Restrict and protect administrator access to the virtualization solution

This is clearly seen as an extension of the process of securing systems that we presented earlier in this chapter.

Hypervisor Security The hypervisor should be secured using a process similar to that with securing an operating system. That is, it should be installed in an isolated environment, from known clean media, and updated to the latest patch level in order to minimize the number of vulnerabilities that may be present. It should then be configured so that it is updated automatically, any unused services are disabled or removed, unused hardware is disconnected, appropriate introspection

capabilities are used with the guest OSs, and the hypervisor is monitored for any signs of compromise.

Access to the hypervisor should be limited to authorized administrators only, since these users would be capable of accessing and monitoring activity in any of the guest OSs. The hypervisor may support both local and remote administration. This must be configured appropriately, with suitable authentication and encryption mechanisms used, particularly when using remote administration. Remote administration access should also be considered and secured in the design of any network firewall and IDS capability in use. Ideally such administration traffic should use a separate network, with very limited, if any, access provided from outside the organization.

Virtualized Infrastructure Security Virtualized systems manage access to hardware resources such as disk storage and network interfaces. This access must be limited to just the appropriate guest OSs that use any resource. As we noted earlier, the configuration of network interfaces and use of an internal virtual network may present issues for organizations that wish to monitor all network traffic between systems. This should be designed and handled as needed.

Access to VM images and snapshots must be carefully controlled, since these are another potential point of attack.

Hosted Virtualization Security Hosted virtualized systems, as typically used on client systems, pose some additional security concerns. These result from the presence of the host OS under, and other host applications beside, the hypervisor and its guest OSs. Hence there are yet more layers to secure. Further, the users of such systems often have full access to configure the hypervisor, and to any VM images and snapshots. In this case, the use of virtualization is more to provide additional features, and to support multiple operating systems and applications, than to isolate these systems and data from each other, and from the users of these systems.

It is possible to design a host system and virtualization solution that is more protected from access and modification by the users. This approach may be used to support well-secured guest OS images used to provide access to enterprise networks and data, and to support central administration and update of these images. However, there will remain security concerns from possible compromise of the underlying host OS, unless it is adequately secured and managed.

12.9 RECOMMENDED READING

[SCAR08] provides general guidance on general server security, which we have closely followed in the chapter. [SCAR11] provides guidance on securing virtualized systems.

SCAR08 Scarfone, K.; Jansen, W.; and Tracy, M. *Guide to General Server Security*, NIST Special Publication 800-123, July 2008.

SCAR11 Scarfone, K.; Souppaya, M.; and Hoffman, M. *Guide to Security for Full Virtualization Technologies*, NIST Special Publication 800-125, January 2011.

12.10 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

access controls	full virtualization	native virtualization
administrators	guest OS	patches
application virtualization	hardening	patching
archive	hosted virtualization	permissions
backup	hypervisor	virtual machine monitor
chroot	logging	virtualization

Review Questions

12.1 What are the basic steps needed in the process of securing a system?
12.2 What is the aim of system security planning?
12.3 What are the basic steps needed to secure the base operating system?
12.4 Why is keeping all software as up to date as possible so important?
12.5 What are the pros and cons of automated patching?
12.6 What is the point of removing unnecessary services, applications, and protocols?
12.7 What types of additional security controls may be used to secure the base operating system?
12.8 What additional steps are used to secure key applications?
12.9 What steps are used to maintain system security?
12.10 Where is application and service configuration information stored on Unix and Linux systems?
12.11 What type of access control model do Unix and Linux systems implement?
12.12 What permissions may be specified, and for which subjects?
12.13 What commands are used to manipulate extended file attributes access lists in Unix and Linux systems?
12.14 What effect do set user and set group permissions have when executing files on Unix and Linux systems?
12.15 What is the main host firewall program used on Linux systems?
12.16 Why is it important to rotate log files?
12.17 How is a chroot jail used to improve application security?
12.18 Where are two places user and group information may be stored on Windows systems?
12.19 What are the major differences between the implementations of the discretionary access control models on Unix and Linux systems and those on Windows systems?
12.20 What are mandatory integrity controls used for in Windows systems?
12.21 On Windows, which privilege overrides all ACL checks, and why?
12.22 Where is application and service configuration information stored on Windows systems?
12.23 What is virtualization?
12.24 What virtualization alternatives do we discuss securing?
12.25 What are the main security concerns with virtualized systems?
12.26 What are the basic steps to secure virtualized systems?

Problems

12.1 State some threats that result from a process running with administrator or root privileges on a system.

12.2 Set user (setuid) and set group (setgid) programs and scripts are a powerful mechanism provided by Unix to support "controlled invocation" to manage access to sensitive resources. However, precisely because of this it is a potential security hole, and bugs in such programs have led to many compromises on Unix systems. Detail a command you could use to locate all set user or group scripts and programs on a Unix system, and how you might use this information.

12.3 Why are file system permissions so important in the Linux DAC model? How do they relate or map to the concept of "subject-action-object" transactions?

12.4 User "ahmed" owns a directory, "stuff," containing a text file called "ourstuff.txt" that he shares with users belonging to the group "staff." Those users may read and change this file, but not delete it. They may not add other files to the directory. Others may neither read, write, nor execute anything in "stuff." What would appropriate ownerships and permissions for both the directory "stuff" and the file "ourstuff.txt" look like? (Write your answers in the form of "long listing" output.)

12.5 Suppose you operate an Apache-based Linux Web server that hosts your company's e-commerce site. Suppose further that there is a worm called "WorminatorX," which exploits a (fictional) buffer overflow bug in the Apache Web server package that can result in a remote root compromise. Construct a simple threat model that describes the risk this represents: attacker(s), attack-vector, vulnerability, assets, likelihood of occurrence, likely impact, and plausible mitigations.

12.6 Why is logging important? What are its limitations as a security control? What are pros and cons of remote logging?

12.7 Consider an automated audit log analysis tool (e.g., swatch). Can you propose some rules which could be used to distinguish "suspicious activities" from normal user behavior on a system for some organization?

12.8 What are the advantages and disadvantages of using a file integrity checking tool (e.g., tripwire). This is a program which notifies the administrator of any changes to files, on a regular basis? Consider issues such as which files you really only want to change rarely, which files may change more often, and which change often. Discuss how this influences the configuration of the tool, especially as to which parts of the file system are scanned, and how much work monitoring its responses imposes on the administrator.

12.9 Some have argued that Unix/Linux systems reuse a small number of security features in many contexts across the system, while Windows systems provide a much larger number of more specifically targeted security features used in the appropriate contexts. This may be seen as a trade-off between simplicity and lack of flexibility in the Unix/Linux approach, against a better targeted but more complex and harder to correctly configure approach in Windows. Discuss this trade-off as it impacts on the security of these respective systems, and the load placed on administrators in managing their security.

12.10 It is recommended that when using BitLocker on a laptop, the laptop should not use standby mode, rather it should use hibernate mode. Why?