Computer Security: Principles and Practice

Chapter 12: Operating System Security

OS Security Layers

- Each layer is vulnerable to attack from below if the lower layers are not secured appropriately

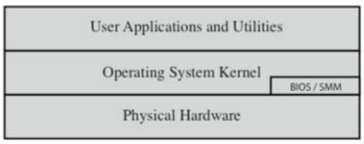
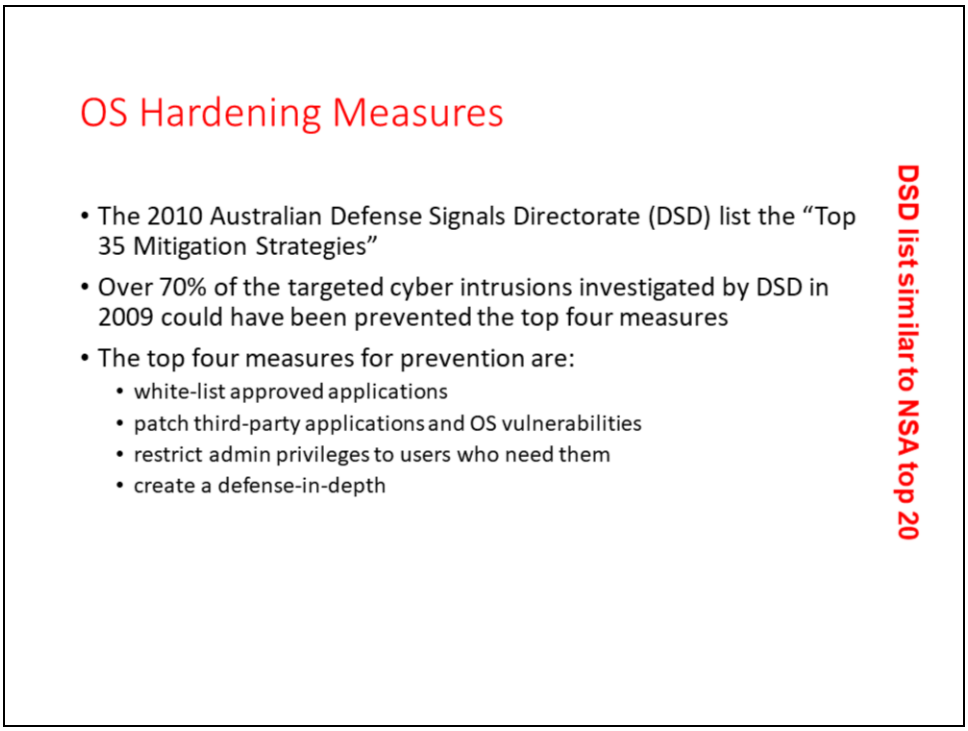


Figure 12.1 Operating System Security Layers

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 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. The 2010 Australian Defense Signals Directorate (DSD) list of the "Top 35 Mitigation Strategies" notes that implementing just the top four of these would have prevented over 70% of the targeted cyber intrusions investigated by DSD in 2009. These top four measures are:

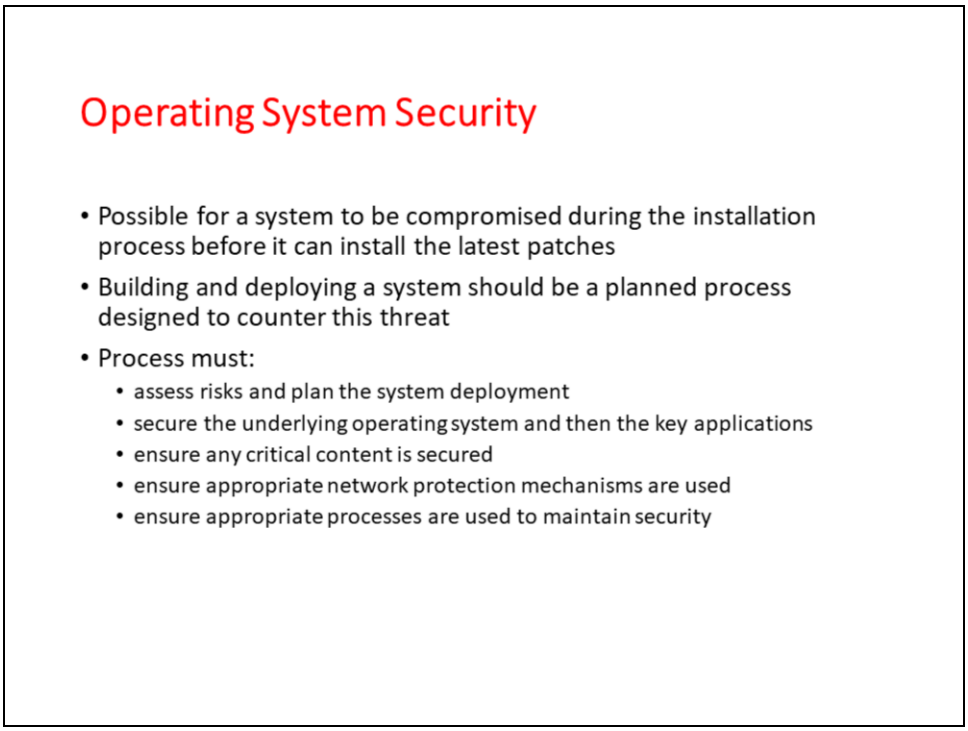
1. patch operating systems and applications using auto-update

2. patch third-party applications

3. restrict admin privileges to users who need them

4. white-list approved applications

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



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.

[NIST08] states that this process must:

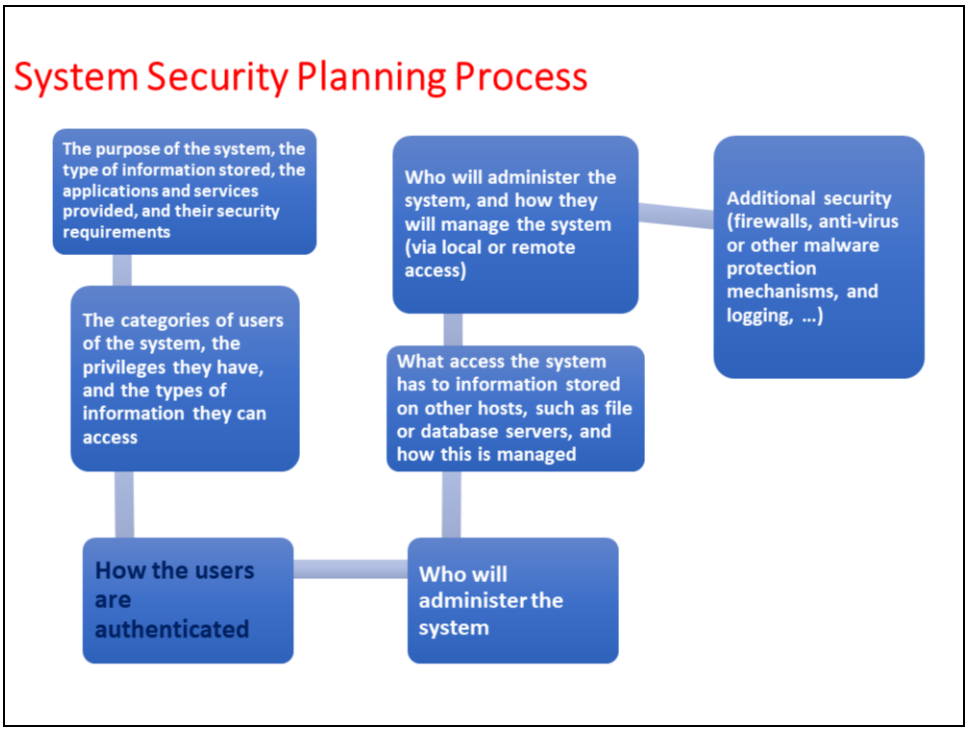
• assess risks and plan the system deployment

- 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.

System Security Planning

- The first step in deploying a new system is planning
  - Plan needs to identify appropriate personnel and training to install and manage the system
  - Planning process needs to determine security requirements for the system, applications, data, and users
- Aim: maximize security while minimizing costs



[NIST08] 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

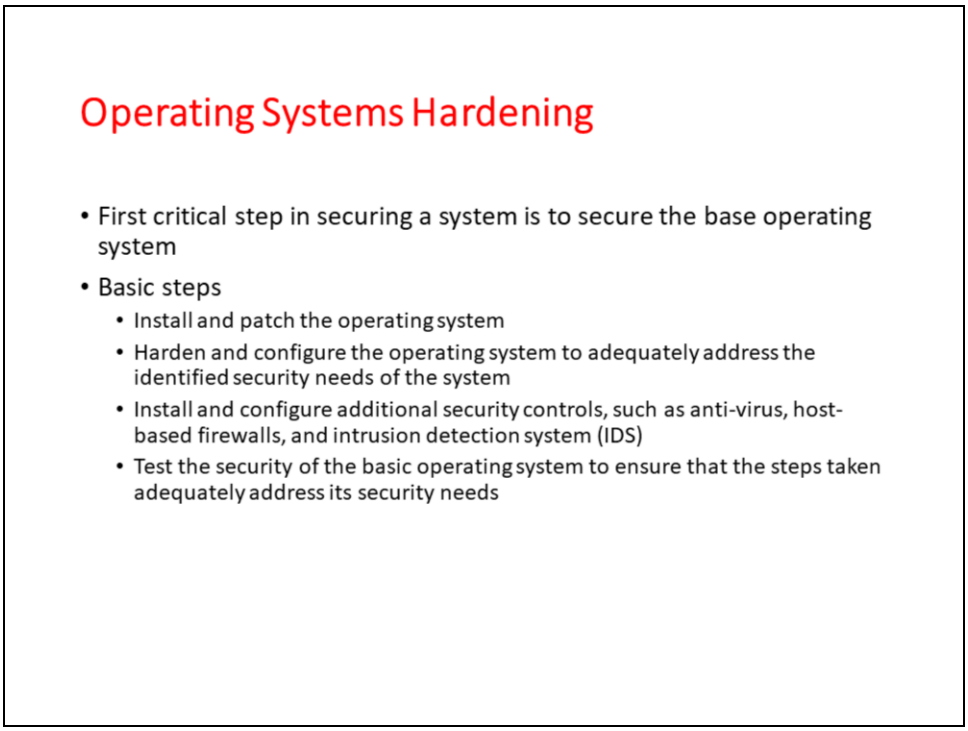
• how the users are authenticated

• 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



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.

[NIST08] suggests the following basic steps 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

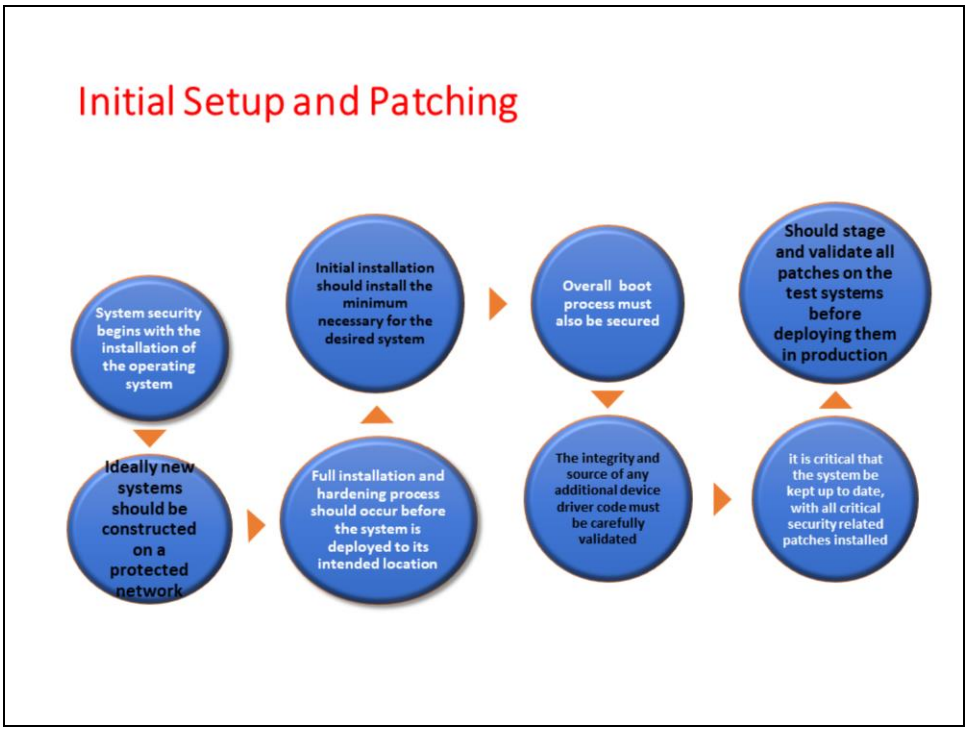
• configuring resource controls

• 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



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.

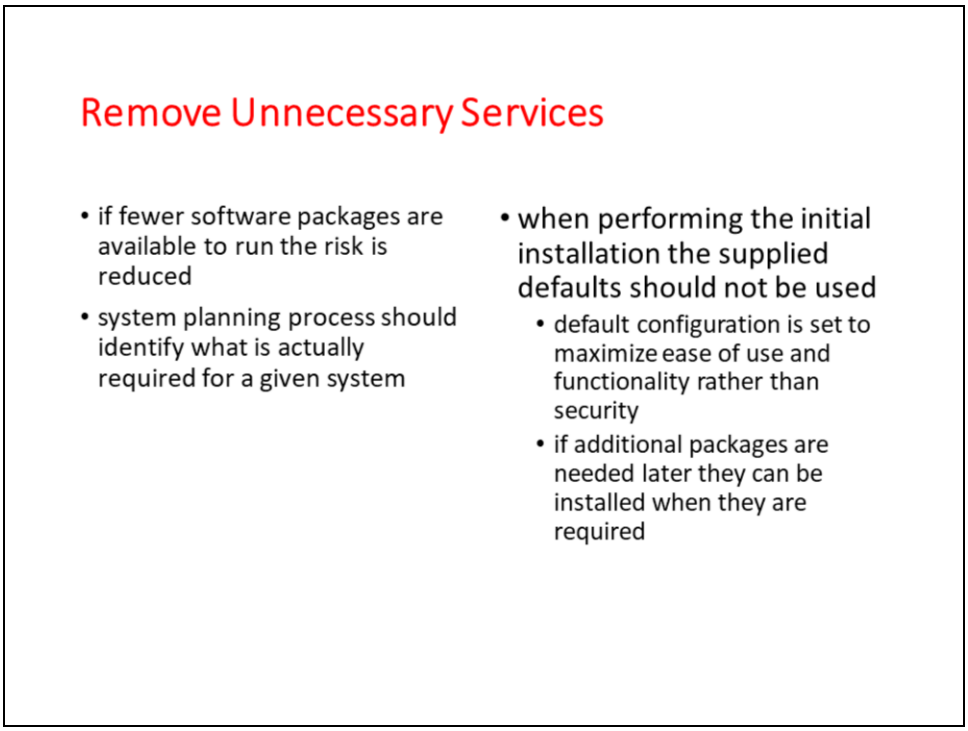
The overall boot process must also be secured. This may require adjusting
options on, or specifying a password required for changes to, the BIOS code used
when the system initially boots. It may also require limiting which media the system
is normally permitted to boot from. This is necessary to prevent 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 the top two of the four key DSD 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.

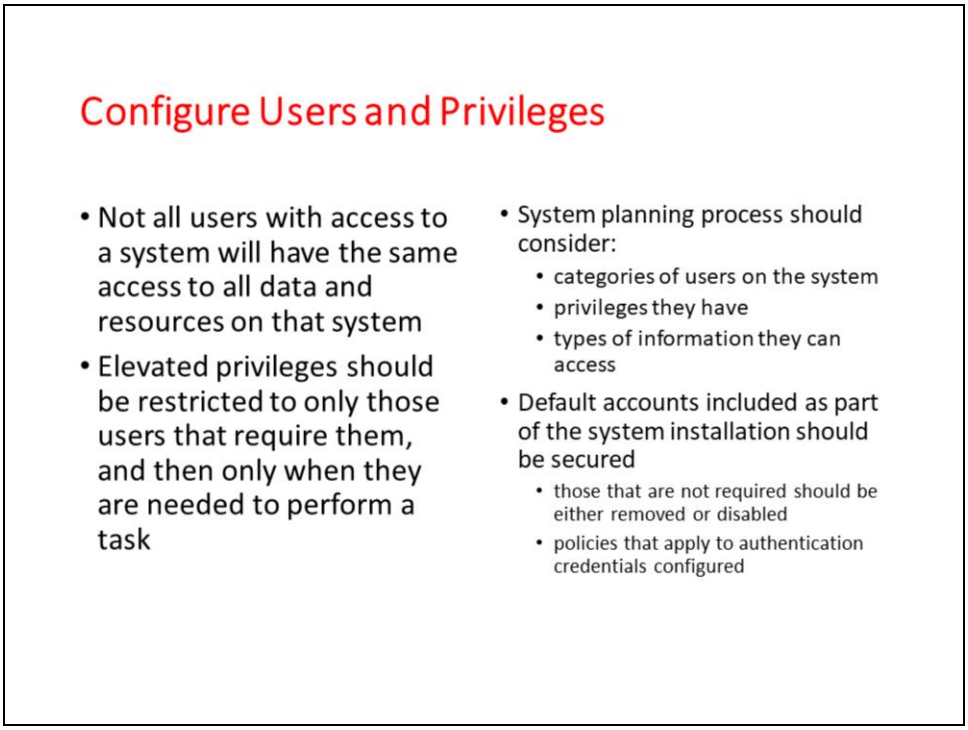
Note that on change-controlled systems, you should not run automatic updates, because security patches can, on rare but significant occasions, introduce instability. For systems on which availability and uptime are of paramount importance, therefore, you should stage and validate all patches on test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



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 required. [NIST08] and many of the security hardening guides provide lists of services, applications, and protocols that should not be installed if not required.

[NIST08] 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.



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 DSD 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.

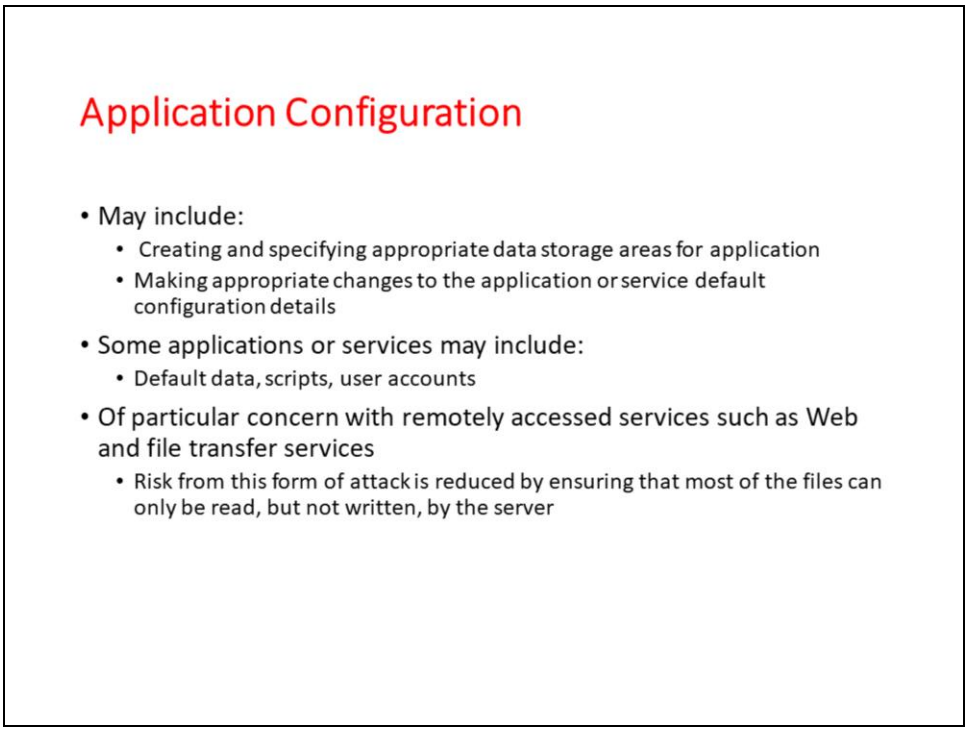
Any policy that applies to authentication credentials, and especially to password security, is also configured. This includes details of which authentication methods are accepted for different methods of account access. And it includes details of the required length, complexity, and age allowed for passwords.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

Configure Resource Controls

- Once the users and groups are defined, appropriate permissions can be set on data and resources
- Many of the security hardening guides provide lists of recommended changes to the default access configuration
- Further security possible by installing and configuring additional security tools:
  - Anti-virus software
  - Host-based firewalls
  - IDS or IPS software
  - Application white-listing

System Testing

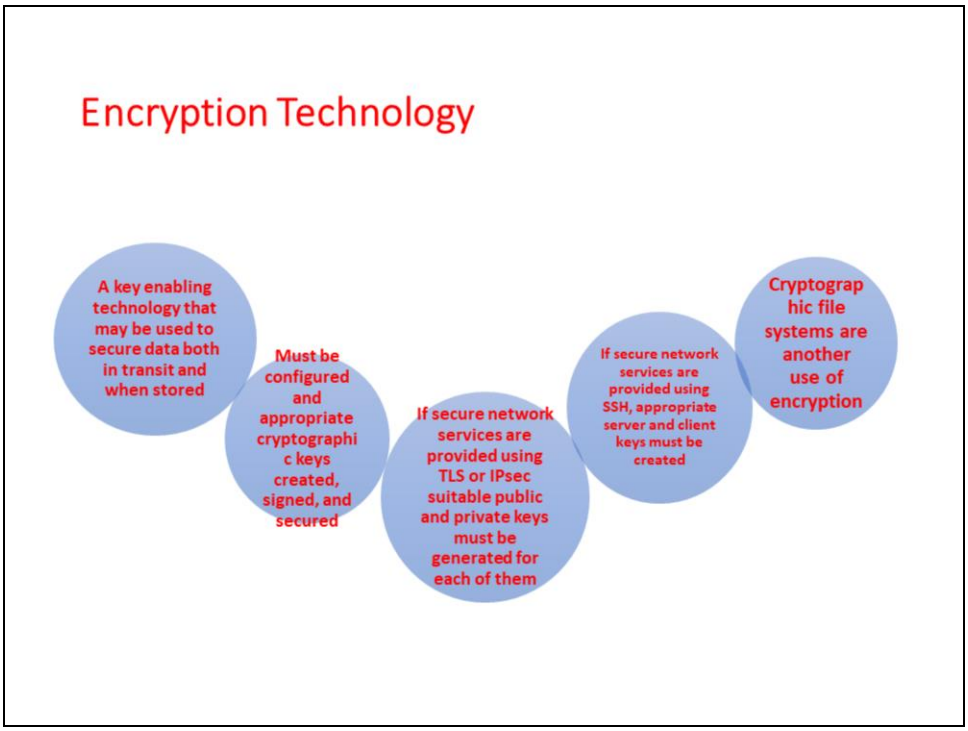
- Final step in the process of initially securing the base operating system is security testing
  - Goal: Ensure the previous security configuration steps are correctly implemented
- Checklists are included in security hardening guides
- There are programs specifically designed to:
  - Review a system to ensure that a system meets the basic security requirements
  - Scan for known vulnerabilities and poor configuration practices



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.

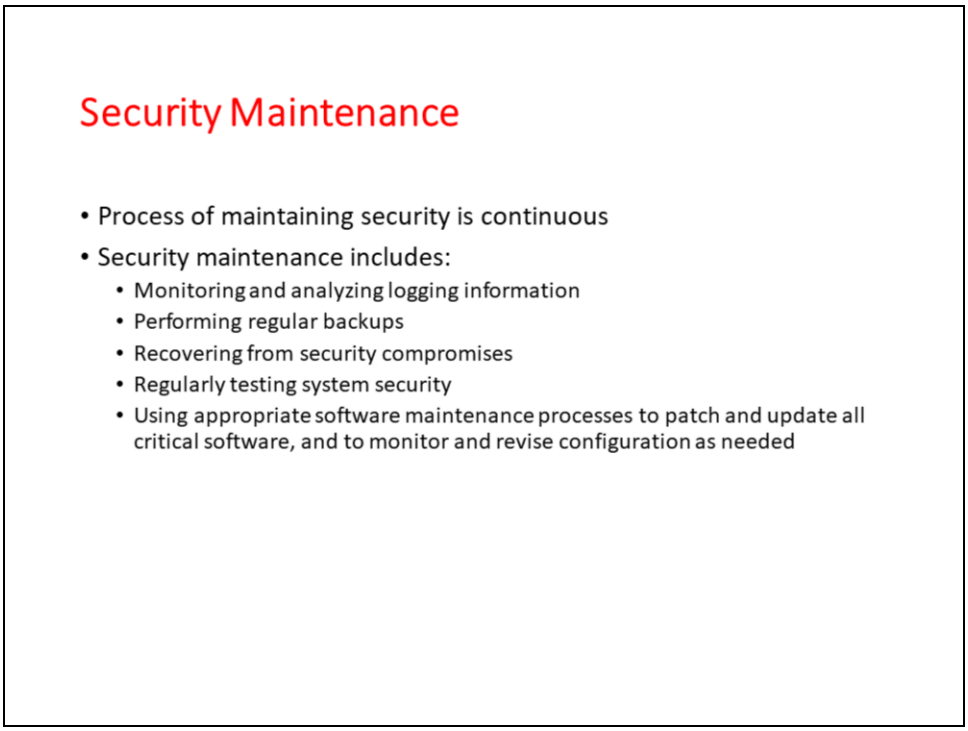
As part of the configuration process, careful consideration should be given to the access rights granted to the application. Again, this is of particular concern with remotely accessed services, such as Web and file transfer services. The server application should not be granted the right to modify files, 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 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.



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. [NIST08] suggests that this process of security maintenance includes the following additional steps:

• monitoring and analyzing logging information

• performing regular backups

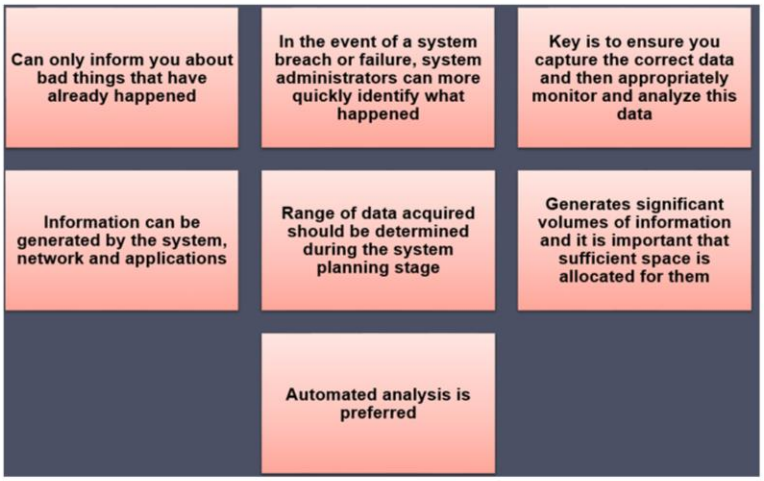
• recovering from security compromises

• regularly testing system security

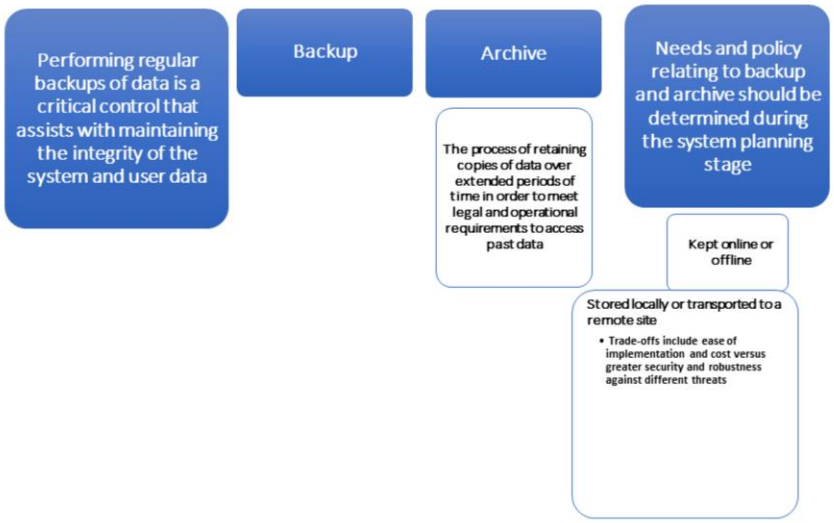
• using appropriate software maintenance processes to patch and update all critical software, and to monitor and revise configuration as needed

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


Can only inform you about bad things that have already happened
In the event of a system breach or failure, system administrators can more quickly identify what happened
Key is to ensure you capture the correct data and then appropriately monitor and analyze this data
Information can be generated by the system, network and applications
Range of data acquired should be determined during the system planning stage
Generates significant volumes of information and it is important that sufficient space is allocated for them
Automated analysis is preferred

Data Backup and Archive



Linux/Unix Security: Patch/Configs

- Patch management
  - keeping security patches up to date is a widely recognized and critical control for maintaining security
  - application and service configuration
  - most commonly implemented using separate text files for each application and service
  - generally located either in the /etc directory or in the installation tree for a specific application
  - individual user configurations that can override the system defaults are located in hidden "dot" files in each user's home directory
  - most important changes needed to improve system security are to disable services and applications that are not required

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.

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 .

Linux/Unix Security

- Users, groups, and permissions
  - access is specified as granting read, write, and execute permissions to each of owner, group, and others for each resource
  - guides recommend changing the access permissions for critical directories and files
  - local exploit
    - software vulnerability that can be exploited by an attacker to gain elevated privileges
  - remote exploit
    - software vulnerability in a network server that could be triggered by a remote attacker

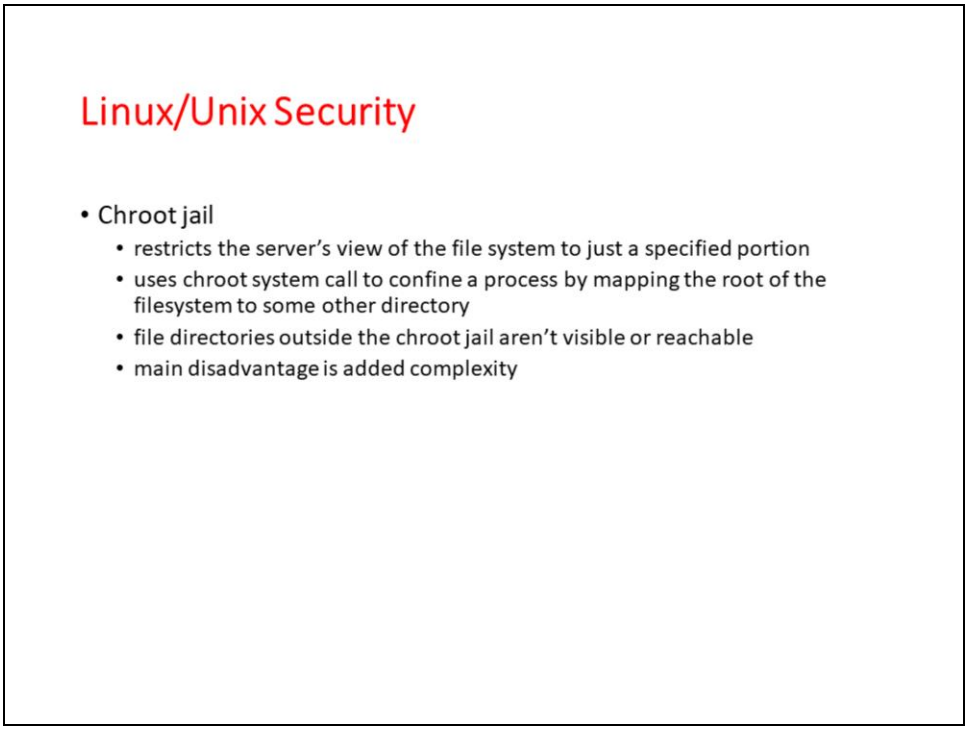
As we describe in Sections 4.5 and 25.3 , Unix and Linux systems implement discretionary access control to all file system resources. These include not only files and directories but devices, processes, memory, and indeed most system resources. Access is specified as granting read, write, and execute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



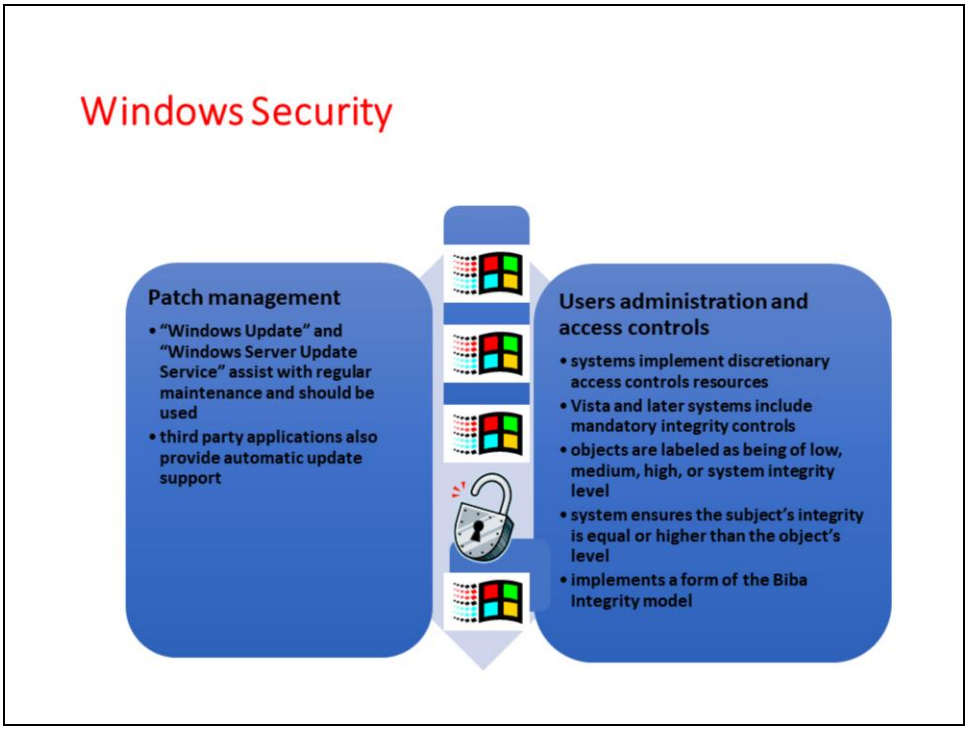
Some network accessible services do not require access to the full file-system, but rather only need a limited set of data files and directories for their operation. FTP is a common example of such a service. It provides the ability to download files from, and upload files to, a specified directory tree. 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.) aren't 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.

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.



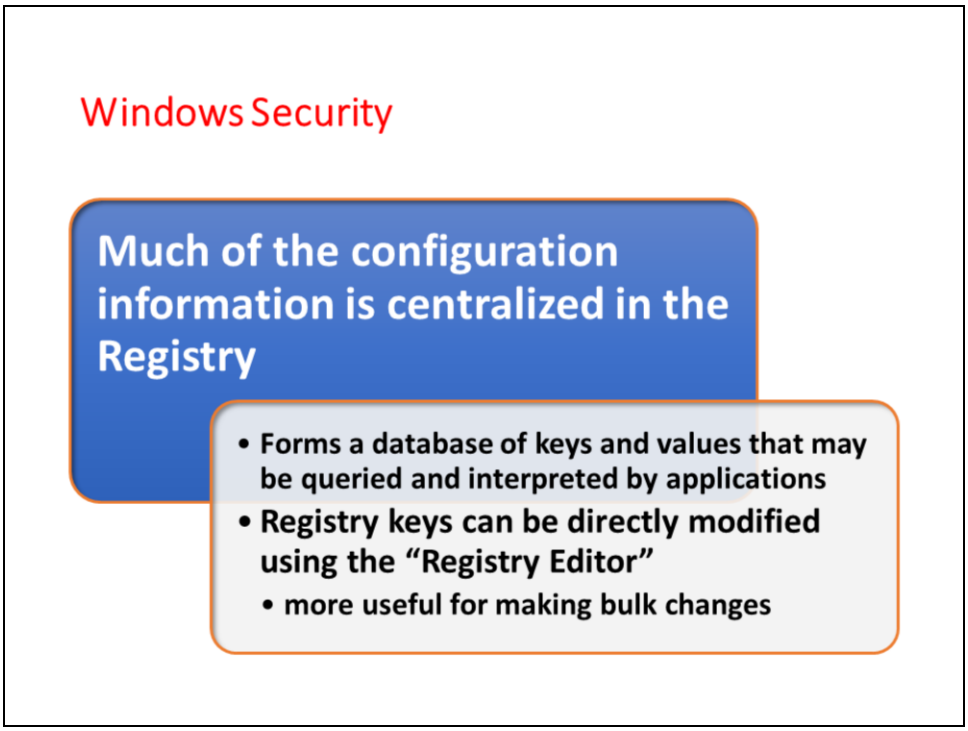
We now consider some specific issues with the secure installation, configuration, and management of Microsoft Windows systems. These systems have for many years formed a significant portion of all "general purpose" system installations. Hence, they have been specifically targeted by attackers, and consequently 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."

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 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 .

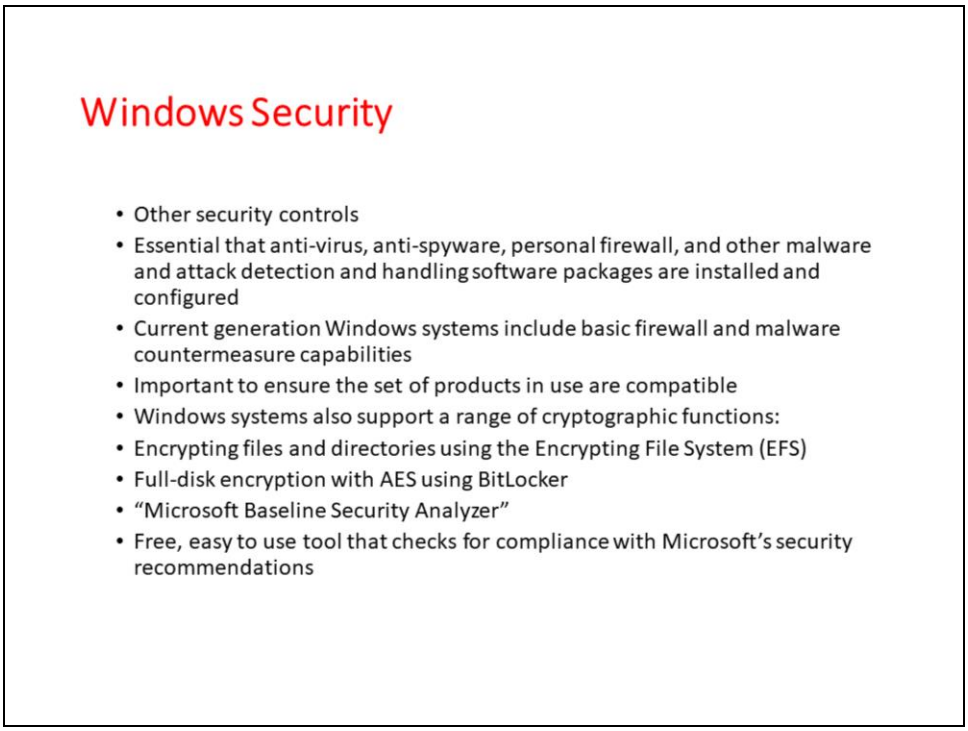
Windows systems implement discretionary access controls to system resources such as files, shared memory, and named pipes. The access control list has a number of entries that may grant or deny access rights to a specific SID, which may be for an individual user or for some group of users. Windows Vista 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.



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.

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 [SYMA11]. However, as the Stuxnet attacks in 2010 show, even isolated systems updated using removable media are vulnerable, and thus must also be protected.

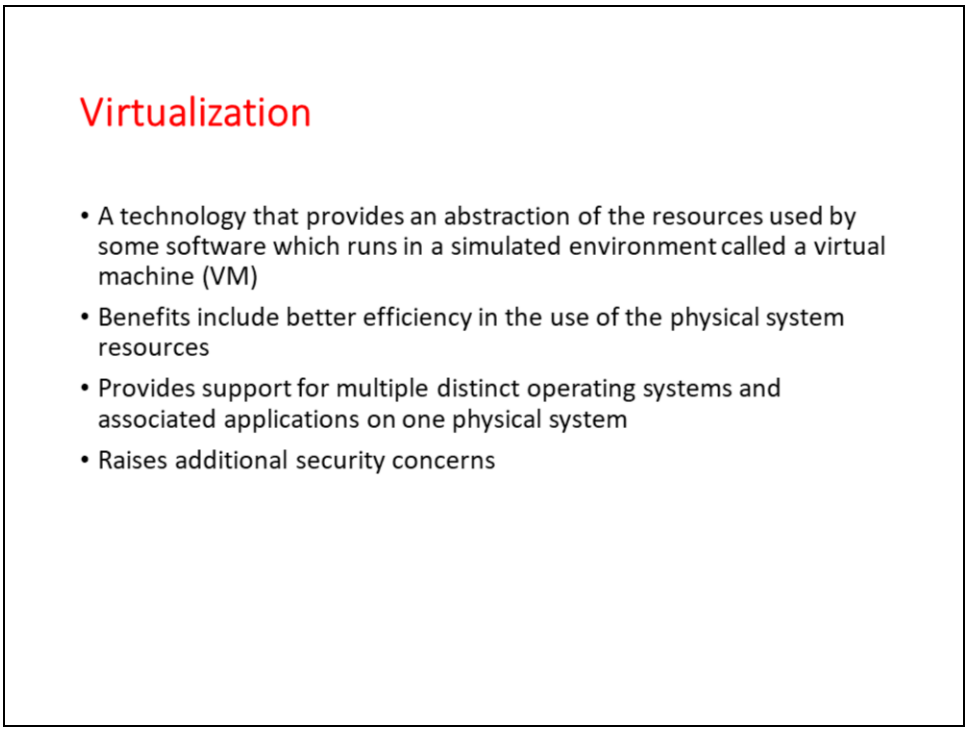


Current generation Windows systems include some basic firewall and malware countermeasure capabilities, which should certainly be used at a minimum. However, many organizations find that these should be augmented with one or more of the many commercial products available. One issue of concern is undesirable 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.

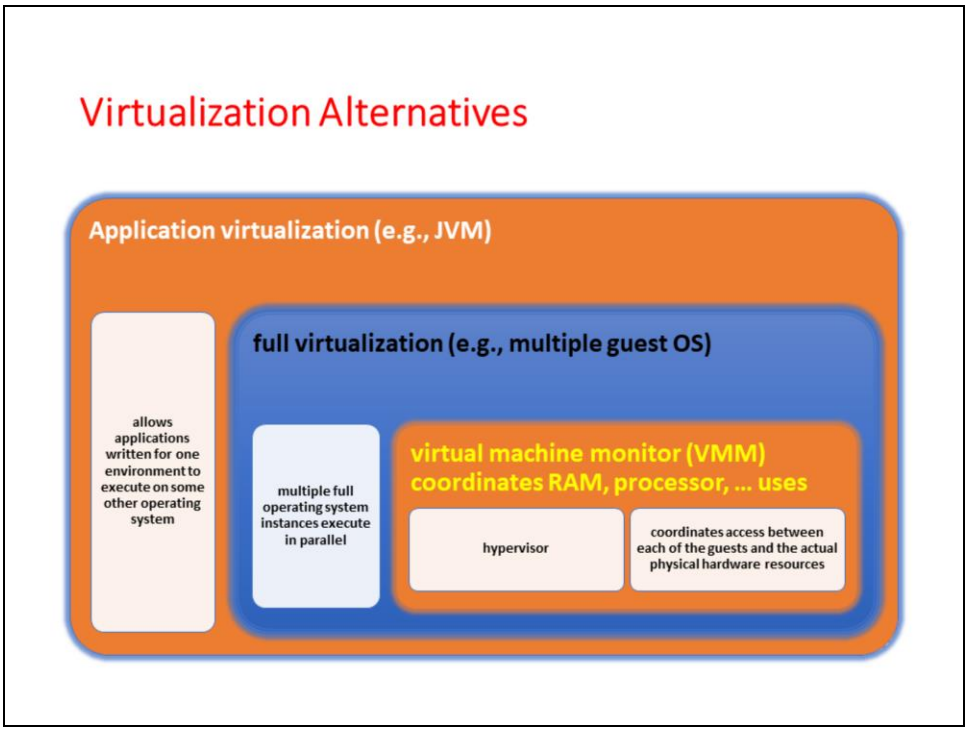
The system hardening guides such as those provided by the "NSA—Security Configuration Guides" also include security checklists for various versions of Windows.

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.



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.

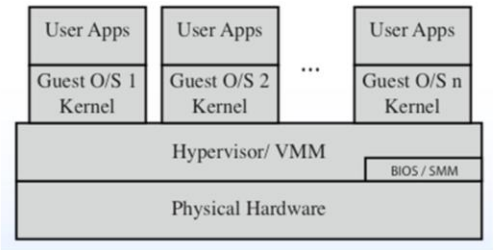


There are many forms of creating a simulated, virtualized environment. These include **application virtualization , as provided by the Java Virtual Machine environment.**

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.

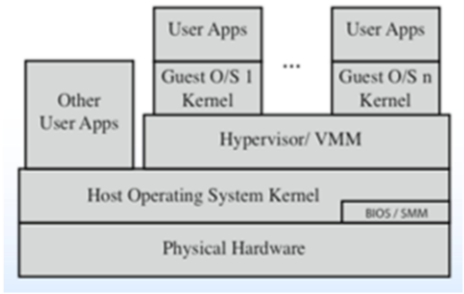
Full Virtualization Variations

- **Native virtualization**: the hypervisor executes directly on the underlying hardware
- Hosted OS is just another app
- More secure: fewer layers



Full Virtualization Variations

- **Hosted virtualization**: Hosted OS run along other apps
- Adds additional layers: increased security concerns

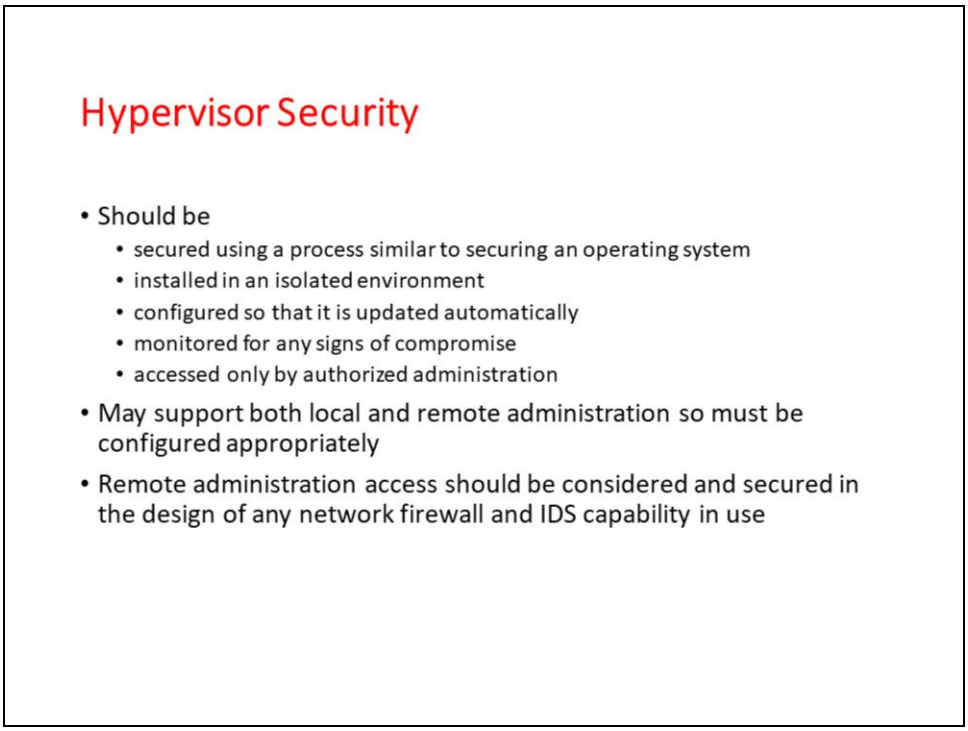


Virtualization Security Issues

- Security concerns include:
  - **Guest OS isolation**: ensuring that programs executing within a guest OS may only access and use the resources allocated to it
  - Guest OS monitoring by the **hypervisor**: has privileged access to the programs and data in each guest OS and ***must be trust***
  - Virtualized environment security: particularly **image** and **snapshot management** which attackers may attempt to view or modify

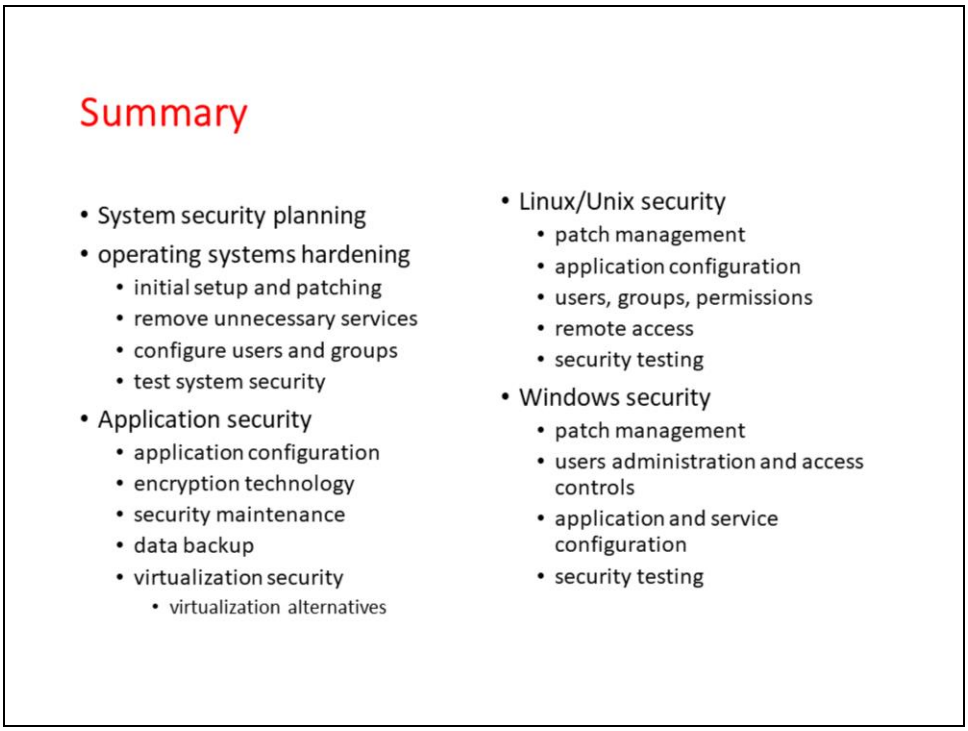
[CLEF09] and [NIST11] both detail a number of security concerns that result from the use of virtualized systems, including:
• guest OS isolation, ensuring that programs executing within a guest OS may only access and use the resources allocated to it, and not covertly interact with programs or data either 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 as regards 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.



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.



Chapter 12 summary.