CHAPTER 2

Basic Components of Linux Security

THIS CHAPTER COVERS BASIC COMPONENTS of Linux security. The starting point for securing a Linux system is the kernel. The kernel is not only the software that runs the system by interfacing between the software and the hardware, it's also the code that gets executed when the system starts. In addition to the kernel and the boot process that helps to protect it, this chapter also looks at potential vulnerabilities associated with Linux applications. Linux security also depends on the authentication databases that apply to users and groups. User security forms the foundation for file ownership and access controls.

As security threats continue, it's important to keep up to speed with the latest Linux security updates. Updates can be a challenge if you administer a large number of systems, especially if they're virtual machines. Finally, if you administer systems based on different Linux distributions, you need to be aware of the differences between the major Linux distributions. A **Linux distribution** is a unified collection of applications, services, drivers, and libraries configured with a Linux kernel. Each Linux distribution is managed by either a company or a group of volunteers.

Chapter 2 Topics

This chapter covers the following topics and concepts:

- How Linux security relates to the kernel
- How you can better secure a system during the boot process
- Linux security issues that exist beyond the operating system
- Which user authentication databases are available for Linux
- How files can be protected with ownership, permissions, and access controls
- How to use firewalls and mandatory access controls in a layered defense
- How to use encrypted communications to protect Linux networks
- How you can track the latest Linux security updates
- The effect of virtualization on security
- How distributions vary

Chapter 2 Goals

When you complete this chapter, you will be able to:

- Describe security features, starting with the boot process
- Understand basic security options for users, groups, and files
- Work with basic security features for networks
- Review appropriate resources for the latest security updates

Linux Security Relates to the Kernel

This section reviews the basic philosophy behind the **Linux kernel** along with the types of kernels that are available. (The Linux kernel is the core of the Linux operating system. Different Linux kernels are in effect different operating systems. The Linux kernel includes a monolithic core and modular components.) To that end, you'll learn about the differences between the stock kernels released by Linux developers and how developers of a distribution modify them.

Frequently, the solution to a security issue is the installation of a new kernel. Although the differences between kernels are often minor, the work required to install a kernel is the same. If you choose to use a custom kernel, it may better serve your needs. However, it takes a lot more work to customize and compile a kernel. You also run the risk of ending up with a nonfunctional operating system if you aren't careful or you don't know what you are doing. The Linux kernel comes with a lot of components. If you leave a critical component out, the kernel will compile but you won't be able to boot the system. Furthermore, when a security issue requires a kernel **patch**, you'll have to repeat the process of customizing and recompiling a kernel. In the context of the Linux kernel, a patch is an incremental upgrade. It is a small piece of source code that replaces existing source code. It is used to fix bugs.

FYI

Whether to choose a **monolithic kernel** or a **modular kernel** is a matter of debate. If you combined all kernel modules into a monolithic kernel, it would load all at once and wouldn't require modules to load as they were needed. Everything would already be loaded in memory, ready to go. On the other hand, operating-system monolithic kernels may be huge, as they include drivers for every conceivable kind of hardware and lots of additional software. A monolithic kernel has all of the configured drivers and features built directly into a single kernel file. A modular kernel has drivers and features in separate kernel module files, separate from the kernel file proper. Modular kernels are often easier to deal with from a development standpoint because the kernel modules can be loaded and unloaded as necessary.

The Basic Linux Kernel Philosophy

The Linux kernel is robust. One reason for this is its structure. It includes a core and supplemental modular components. The monolithic part of the kernel contains those drivers and options essential to the kernel boot process. Any modular part of the kernel adds everything else needed for smooth operation, including many drivers that enable communication with installed hardware.

For example, most Linux sound cards are connected with drivers that are loaded with kernel modules if and as needed. If that kernel module fails, it does not affect other parts of the Linux operating system. You may just have to live without sound until you can load or reload a working driver. macOS uses an even smaller kernel than Linux does. This is because macOS is based on Mach, which was designed to be a very small and fast kernel. Windows is a mix of monolithic and modular, as Windows supports kernel mode modules, just as Linux does.

Some central processing units (CPUs) enforce protection rings, requiring the highest level of permissions to gain access to hardware. One of the primary functions of the kernel is to manage and interface with the hardware, which means there is at least portions of the kernel require those high levels of permissions. While Intel-based hardware supports multiple protection rings, not all operating systems make use of the different levels of permissions. On Linux, two protection rings are used. The first is supervisor mode, which is where the kernel runs. While drivers and loadable modules might use slightly lower permission sets in other operating system, Linux runs every module in kernel space with ring 0 permissions, which is the innermost ring in this model and the one with the highest permission set. Everything else runs in user mode, which has no permissions to the hardware. If a user program needs to interface with the hardware, it has to send make system calls to the kernel, while the kernel talks to the hardware on behalf of the user program.

Basic Linux Kernels

Most Linux kernel developers are volunteers. Only a few are affiliated with a specific distribution such as Red Hat or Ubuntu. They follow open-source principles, which means the source code to the kernel is freely available to download and inspect. Development work is managed by the **Linux Kernel Organization**. You can track news and other kernel-related activity at https://linux.org. The source code for the kernel can be acquired at https://kernel.org. The Linux Kernel Organization is a nonprofit group established to distribute the Linux kernel and other open-source software. All Linux-based distributions use kernels that have been developed by the Linux Kernel Organization. This is not to say all kernels are the same. It is up to the developers of the distribution to create their own Linux kernel for release in the distribution. This means they determine the configuration of the kernel and use that configuration to build binary software.

Building a custom kernel for yourself requires starting with Linux **source code**. This is the same source code used by Linux distributions, but you end up with a base kernel and any modules in any configuration you specify.

When bugs or vulnerabilities are discovered in the kernel, developers may issue fixes in the form of patches. A patch is a differential between the main source code and changes that fix the bug or vulnerability. A patch is applied to the source code, resulting in updated source. A patch may be released if there isn't a release scheduled or deemed necessary. The patch means you can apply the individual fix to an existing source code tree without having to grab a completely new version. Each new version of the Linux source code will offer a patch, which provides the differences between the last version and changes that are in the newest release.

Today, you can grab the latest kernel source code using the source code management program git, which was developed specifically for managing development of the Linux kernel across a large number of developers and needs of the overall kernel. The way the kernel is organized today, there is no single repository for the entire kernel. Instead, you can view different components from https://git.kernel.org, which will point you at the git file that would allow you to pull that portion of the overall source code. More commonly, you would grab a tarball of the latest release, since there is no guarantee of the status of any of the individual branches of the source code. The released tarball would have gone through testing to ensure all the different elements build together and work in a built configuration.

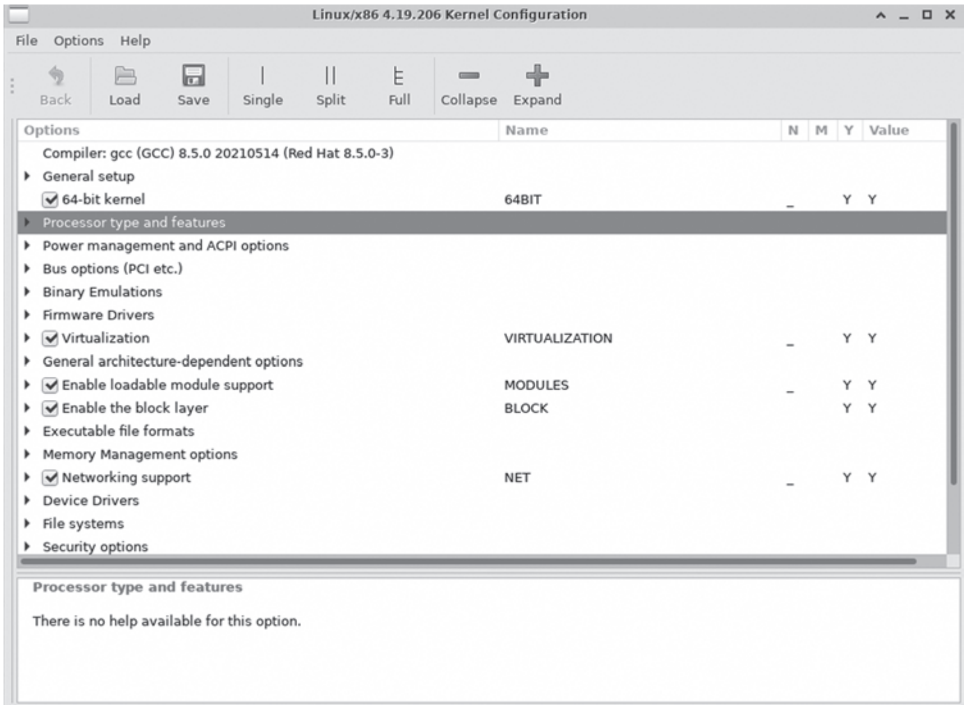
Distribution-Specific Linux Kernels

For most users, it's best to stick with the **binary kernel** released for a Linux distribution. That distribution-specific kernel has been compiled and built with the intended hardware in mind, and in many cases has been labeled as such. This is especially true when it comes to 64-bit versus 32-bit hardware. While modern processors are commonly 64-bit today, there are still plenty of systems that are or could be running 32-bit since Linux systems often don't have heavy hardware requirements. The name of the kernel file will generally include the version number as well as the hardware architecture. For example, one of the latest releases of Debian Linux has a kernel named vmlinuz-5.10.0-8-amd64. The last part, amd64, indicates the architecture, meaning it's 64-bit. Even if the kernel is not so labeled, it may be optimized for certain purposes—for example, high availability or database support.

When security issues appear with a Linux kernel, most popular Linux distributions release updates fairly quickly. They've done the work to compile the kernel for you. Their compiled kernel packages will even automatically update standard Linux boot loaders. All you need to do is download and install the updated kernel. This is usually done by keeping your distribution up-to-date using the native package management tool. In the case of Debian, this would be apt.

Custom Linux Kernels

If you choose to customize a Linux kernel, you'll need to get into its nuts and bolts. The Linux kernel is highly customizable. The 4.18 kernel line, included with Red Hat Enterprise Linux 8, has more than 5,000 configuration options. If you want to preview some of these options, look for a config-* file in the /boot directory of a Linux system. It's a text file. These options are divided into a number of categories. One standard customization menu with some of these categories is shown in **Figure 2-1**. There are different configuration interfaces, including a text-based one using the ncurses library. Figure 2-1 shows the GTK-based configuration interface. This allows you to work


Linux/x86 4.19.206 Kernel Configuration
File Options Help
Back Load Save Single Split Full Collapse Expand

```
Options                                          Name              N  M  Y  Value
  Compiler: gcc (GCC) 8.5.0 20210514 (Red Hat 8.5.0-3)
▸ General setup
  ☑ 64-bit kernel                                64BIT             _     Y  Y
▸ Processor type and features
▸ Power management and ACPI options
▸ Bus options (PCI etc.)
▸ Binary Emulations
▸ Firmware Drivers
▸ ☑ Virtualization                               VIRTUALIZATION    _     Y  Y
▸ General architecture-dependent options
▸ ☑ Enable loadable module support               MODULES           _     Y  Y
▸ ☑ Enable the block layer                       BLOCK                   Y  Y
▸ Executable file formats
▸ Memory Management options
▸ ☑ Networking support                           NET               _     Y  Y
▸ Device Drivers
▸ File systems
▸ Security options

Processor type and features

There is no help available for this option.
```

Courtesy of Linux Foundation.

FIGURE 2-1 A Linux kernel configuration menu.

through the multiple layers of configuration options, checking boxes where you have binary (on/off) choices for options.

Table 2-1 provides an overview of the categories of options associated with the Linux kernel. As kernel development continues, these categories may change.

Take some time to understand the options associated with the Linux kernel. It's worth the trouble. Even if you never recompile a kernel, that knowledge will help you understand how the kernel can help maintain Linux security. In addition, some of those options can be changed in real time, with the help of options in the /sys/ directory and using the sysctl utility as well as the /etc/sysctl.conf configuration file.

If you choose to recompile a kernel, you'll need to download the source code. In general, it's best to start from one of two different source-code trees. You can start with the kernel source code directly from the Linux developers. Alternatively, you can start with the source code released by a distribution, which may include settings related to the distribution. As the source code for all Linux kernels are freely available under

WARNING

The programming libraries required to compile the Linux kernel can present security risks. If a malicious user gets access to that system, that person can run those compilers to build more dangerous malware. If you choose to compile a custom kernel, you need to either compile that kernel on a system isolated for that purpose or uninstall those libraries after the compile is complete.

TABLE 2-1 Basic configuration categories for the Linux kernel.

CATEGORY	DESCRIPTION
General setup	Includes a variety of basic kernel settings
Loadable module support	Sets conditions for how modules are loaded for a modular kernel
Block layer	Defines how block devices can be loaded
Processor type and features	Configures support for a variety of processor types and memory levels
Power management options	Defines available power-management features
Bus options	Specifies support for hardware cards
Binary emulations	Determines whether additional binary types may be supported, including 32-bit executables
Executable file formats	Associated with executable and linkable format binaries
Networking support	Includes network drivers, protocols, and more
Device drivers	Support for a wide variety of hardware
Firmware drivers	Supports nonvolatile systems such as the Basic Input/Output System (BIOS) and the Unified Extensible Firmware Interface (UEFI)
Filesystems	Includes various filesystem formats
General architecture-dependent options	Includes settings for features like stack protection
Memory management options	Has settings for how virtual memory is handled, including page sizes and other options
Kernel hacking	Designed for kernel debugging
Security options	Includes options such as Security Enhanced Linux (SELinux) and Application Armor (AppArmor)
Cryptographic API	Defines supported encryption algorithms for application programming interfaces (APIs)
Virtualization	Adds code for hardware-based virtual machines
Library routines	Includes modules for cyclic redundancy checks and decompression support

open-source licenses, the choice is yours. In some cases, grabbing the kernel source directly from the Linux project will be easier, though using the distribution's source, you will start with the options indicated by the distribution developers, so you may end up with fewer broken kernel binaries.

Linux administrators who want the most secure kernel will remove or disable many features. If used by a malicious user, some of these features may lead to security risks. But take care. Test any changes that you make. With thousands of options, some changes could make that kernel unusable.

Linux Kernel Security Options

Linux kernel security options range far and wide. Direct options may enable features such as AppArmor, SELinux, and **iptables**-based firewalls (iptables is the Linux packet filtering command for firewalls and masquerading). A common way systems have been exploited for decades has been through the use of buffer overflow vulnerabilities. The Linux kernel now has protections built in that keep buffer overflows from happening, or at least happening easily. These are some of the configuration settings for kernel-based stack protections.

```
CONFIG_HAVE_STACKPROTECTOR=y

CONFIG_CC_HAS_STACKPROTECTOR_NONE=y

CONFIG_STACKPROTECTOR=y

CONFIG_STACKPROTECTOR_STRONG=y
```

There are different ways to protect against these sorts of attacks, starting with languages that protect against stack abuse. You can also have the compiler introduce stack protection in languages that do normally allow for stack abuse. This is done by introducing a value into the stack that is checked before the return address is jumped to. If the value doesn't match what it is supposed to, the return address is considered corrupted and the program fails. This technique is called a stack canary, so-named after the canary coal miners are said to have taken into the mine as an early warning about fatal gases in the mine. If there is a problem with the return pointer, the stack canary will be killed, indicating the program's memory space has been corrupted. Another approach to protecting application memory is to prevent stack memory from being executed. After all, the stack is intended to be a place where program data known at compile time are stored and retrieved. Kernel settings to enable protections against an executable stack are shown here. You can see these are set as on, meaning the protections have been enabled.

```
CONFIG_ARCH_HAS_STRICT_KERNEL_RWX=y

CONFIG_STRICT_KERNEL_RWX=y
```

The configuration settings here are shown out of the file .config, since all configuration settings are stored in plaintext, regardless of how you perform the configuration. If you prefer, you can just edit the text file and not use the menu system of configuration.

Generally, the menus are easier to work with and will make sure you don't mess with the syntax of the configuration file inadvertently.

Securing a System During the Boot Process

If a malicious user has physical access to a system, that user can boot a fully functional version of Linux on your servers from a **live CD**—with full root administrative access. Even without a live CD, some distributions allow a user to boot Linux in single-user mode with full administrative privileges—without a password. (A live CD is a CD or DVD with a bootable operating system. That same data may also be loaded on a USB drive.)

This chapter will assume that the boot process ends once the Linux kernel is loaded on the system. As you'll see later in this chapter, the startup process continues by loading preconfigured services.

Unified Extensible Firmware Interface (UEFI)

It used to be the boot process for systems was handled in the first sector of the primary hard drive of a computer. The problem with this is that the first sector is too small and doesn't allow for large partition sizes or very many boot options. The master boot record (MBR) for partition tables and storing boot code has been replaced by the UEFI. The partition table in the MBR has been replaced by the GUID partition table (GPT), which allows for larger partition sizes and more partitions. It also enables the use of UEFI, which creates more capabilities in a pre-boot environment because the initial boot code can be larger than the 446 bytes a traditional MBR allows.

The challenge with this is it can also introduce capabilities for attackers to insert themselves into the boot process by making adjustments to the boot code being run by the UEFI boot process. Fortunately, UEFI also allows for secure boot, which can ensure the boot process has not been tampered with. Enabling GPT to support UEFI is another set of kernel configuration options, seen here.

```
CONFIG_EFI=y

CONFIG_EFI_STUB=y

CONFIG_EFI_MIXED=y

CONFIG_EFI_VARS=y

CONFIG_EFI_ESRT=y

CONFIG_EFI_VARS_PSTORE=y

CONFIG_EFI_VARS_PSTORE_DEFAULT_DISABLE=y

CONFIG_EFI_RUNTIME_MAP=y
```

Physical Security

Physical security goes beyond locks in the server room. It also involves security for CD/DVD drives and universal serial bus (USB) ports. While a live CD may in some cases be an excellent way to recover from problems on a server or workstation, access to such systems should be limited. Live CDs provide password-free access to the administrative account. Therefore, a malicious user with a live CD may be able to do anything that an administrator can do to your system.

The Threat of the Live CD

It's not enough to set up secure usernames and passwords on your systems. A malicious user with physical access may be able to boot a live CD Linux distribution such as **Knoppix**, Ubuntu, or even **CentOS** directly on your servers. (Knoppix is a Linux distribution best known for its live CDs and DVDs. Short for the Community Enterprise Operating System, CentOS is sometimes known as a rebuild because it is a distribution built by third parties, based on source code released for the Red Hat Enterprise Linux distribution.) In fact, most Linux distributions include a live boot option, usually from the installer disc. It doesn't matter if the Linux distribution on the live CD differs from that on the hard drive. It also doesn't matter if the Linux kernel on the live CD has a different version number. Many live CDs even provide a graphical user interface (GUI), making it relatively easy for even less-experienced users, regardless of their motive, to break into your system.

All that's needed is boot access from a CD/DVD drive or a USB port. Once booted, a live disk (CD/DVD/USB) provides password-free administrative access to any system. At that point, it's easy to access hard drives on the local system.

Boot Process Security

Anyone with physical access to your systems may be able to access boot menus, such as those available through a BIOS or a UEFI menu. Such menus should be password-protected. Network access to the UEFI menu should be disabled. If possible, these menus should be used to exclude all options but the hard drive with the operating system from the boot process.

Boot process security extends to the boot menu for the operating system. Linux uses the **Grand Unified Boot (GRUB)** loader. This is a two-stage boot loader with an initial, small set of boot code. This points to a secondary loader, which loads the actual operating system, or kernel. Unless password protected, the GRUB boot loader can be used to boot a system into single-user mode, with full administrative privileges. On most Linux distributions, an administrative password isn't required to access this mode. The GRUB loader has configuration settings that tell how the kernel image is to be loaded. If you interrupt the boot process, you can edit the configuration and then boot using the new settings. If you take the following configuration, you can add the word *single* to the configuration line and the system will boot into single user mode, providing full administrative privileges to the system.

```
  kernelopts=root=/dev/mapper/cs-root ro crashkernel=auto
resume=/dev/mapper/cs-swap rd.lvm.lv=cs/root rd.lvm.lv=cs/swap
rhgb quiet
```

Other boot loaders can be installed on most systems, including Microsoft options such as NTLDR and bootmgr, as well as third-party options such as BootIt and Partition Magic. Each of these boot loaders can be used to start a Linux operating system.

More Boot Process Issues

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite includes default port numbers for hundreds of services, such as port 22 for Secure Shell (SSH) and port 23 for Telnet. Some security professionals prefer to use nonstandard port numbers for key services. That obscurity can slow the efforts of attackers who want to break into a system.

Some services require network access during the system startup process. For example, if you keep servers synchronized with the **Network Time Protocol (NTP)**, those servers may need to access external NTP servers during the boot process. NTP is a protocol and service for synchronizing clocks across systems on a network. The standard NTP port is 123. If you want to block that port after the boot process is complete, you need to make sure the firewall isn't started until after the NTP client is synchronized. At that point, if you introduce such a firewall rule, you will no longer have the ability to sync with the NTP servers as long as the system is running and the firewall rule is in place.

Virtual Physical Security

Virtual machines add another dimension to the challenges of physical security. Standard Linux user and group options add security to virtual machines. However, anyone who gains access to a virtual machine can more easily change virtual physical components. For example, a malicious user who adds a live CD file to a virtual CD drive would arouse less suspicion than a malicious user who unlocks a CD/DVD drive on an actual physical system. Where physical systems run the risk of a similar situation, virtual machines are even more at risk. An adversary who gets access to the host system can make changes to any virtual machine on that host. Where physical systems require physical access to alter the hardware, virtual machines can be modified through changes to configuration files, and those changes can be made remotely.

Linux Security Issues Beyond the Basic Operating System

While issues have been found within the Linux kernel, the majority of security vulnerabilities within Linux distributions are found within the applications and services that run on top of the kernel.

One of the main advantages of Linux is its structure as a multiuser operating system. When properly configured, Linux services run on nonprivileged accounts. If a malicious user breaks into one service, that user may gain access to the corresponding nonprivileged account. However, that user won't have access to other accounts.

A GUI is also beyond the basic Linux operating system. If included, a GUI introduces a whole raft of additional security risks, discussed shortly.

Service Process Security

For the purposes of this chapter, the boot process ends when the Linux kernel is loaded. The startup process then begins automatically, loading and starting a series of services configured in the **default runlevel**. A **runlevel** defines the services to be run. The default runlevel defines the services to be run after a system is booted up. Most distributions try to keep the number of default services to a minimum, but a couple that you might expect to see are the SSH daemon, sshd—allowing remote administrative connections—and syslog, which is the system logging service.

Red Hat Enterprise Linux includes a number of services that are active by default. Most users do not need many of these services. **Table 2-2** lists a small number of services that were active after installation of a minimal 64-bit server system. On a CentOS 8 system, 42 services are installed and running, though this is on a system that includes a graphical interface, which would not be common for a server installation.

If any service is not maintained, related security flaws may not get fixed. This may be a security flaw that has been located and an exploit developed by an adversary. The malicious user is free to use that exploit as long as the system remains unpatched. Any other attacker can also use exploits that may be found in the wild to attack your system. This is why it's essential to have a patch-management and update process.

Other services that may be active on default Linux installations include the Apache web server, the Samba file server, the Network File System service, the **sendmail** email service, and the **Very Secure FTP** server. (The sendmail email service is the open-source SMTP server maintained by the Sendmail Consortium. Do not confuse this with the commercial SMTP server known as **Sendmail**.) Because security issues on some of these services are relatively common, administrators

NOTE

You specify a default runlevel that will be selected each time the system starts without intervention by someone when the system is booting up. The runlevel determines which services will start during the init process, which is the super process that is responsible for starting all other processes during system startup.

NOTE

The idea of runlevels exists in systems that use systemd instead of init for system initialization, but they are called targets. Some of the targets are the same as runlevels on init systems. When you install a graphical user interface, the target is graphical.target. This tells system to start up the display manager service. By contrast, multi-user.target allows multiple users to login. You can get a list of targets by running *systemctl list-units --type target*.

TABLE 2-2 Some common active services.

SERVICE	DESCRIPTION
cups	**Common Unix Printing System (CUPS)** service
dbus	D-Bus system message bus
ntpd	NTP daemon
sshd	SSH service
syslog	System log message service

should disable these services if they're not being used. In fact, any service that isn't being used should be disabled to limit the attack surface exposed to adversaries.

NOTE

The Very Secure FTP server is a GPL version 2–licensed **File Transfer Protocol (FTP)** server for Unix and Linux. FTP is a standard protocol and service for exchanging files. It is an FTP service that has had some well-known vulnerabilities, though the last one as of this writing was disclosed six years ago.

Security Issues with the GUI

Applications that require a GUI have the potential to introduce vulnerabilities that may be exploited. One reason for this is GUI-based systems have a significantly larger collection of packages that get installed. Programs written for GUIs also have a much larger set of libraries that get added during the build, which means there is more code that can be broken. While most GUI-based systems do not allow for remote logins, it is possible to allow remote logins using a GUI, and some of those remote logins can be highly problematic.

When it comes to servers, a rule of thumb is to reduce the number of packages that are installed in order to reduce the potential for vulnerabilities. Fewer packages mean fewer things that can go wrong. Since Linux administration can very often be handled on the command line, getting rid of the graphical interface should not be a handicap. You can easily install the SSH server to get remote access to a command line interface. As SSH is encrypted, there is no potential for the exposure of usernames and passwords over the network as there was with the Telnet protocol that was formerly used to allow remote, command line access to systems.

Linux User Authentication Databases

NOTE

Since Samba version 4, you can use your Linux system as a Windows domain controller to authenticate Windows users. Samba can also be configured to authenticate to a Windows Active Directory server.

Four major user authentication databases are available for Linux. The local Linux authentication database is known as the shadow password suite. The files of the shadow password suite are /etc/passwd, /etc/group, /etc/shadow, and /etc/gshadow. These files include encrypted passwords, user and group accounts, home directories, and login shells.

The other three user authentication databases are designed as a central authentication database for multiple systems on a network. When properly configured, a user can log into any connected workstation. That workstation checks login credentials from that central database. Two of these databases are native to Linux: the Network Information Service (NIS) and the Lightweight Directory Access Protocol (LDAP). With the help of the **Winbind** service, Linux can also serve as the domain controller repository for some authentication databases commonly used on Microsoft-based networks. Winbind is a component of the Samba file server that supports the integration of Linux/Unix and Microsoft authentication information.

In all these cases, the user and the group get both a user ID and a group ID number. With those ID numbers, each user (and group) can own files on the local Linux system. Each networked user can be configured as a member of other special groups with administrative privileges, just like any other Linux user.

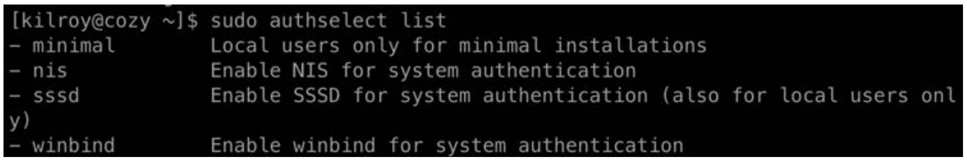
The advantage of NIS is that it can start with the files of the shadow password suite. NIS has two disadvantages, however: It is relatively insecure, and it does not support authentication of Microsoft users or groups. In contrast, while LDAP is a more complex authentication scheme, it does support authentication of both Linux/Unix and Microsoft users and groups.

In a Linux authentication database on a desktop system, a user is typically configured as a member of a group with the same name. For example, if a Linux system has a user named mike, that user is a member of a group named mike. That Linux user mike may be a member of other groups. For example, one group may have print-administration privileges; another group may have access to telephone modems.

To help manage this variety of authentication databases, Linux uses the /etc/nsswitch.conf file, also known as the name service switch file. That file lists databases associated with a number of different configuration files in the /etc/ directory, in some specific order. That order determines where the local system looks first to verify a username and password. The command line tool used on CentOS/RedHat Enterprise Linux for managing authentication sources is *authselect*. You can see the list of authentication sources on a CentOS system in **Figure 2-2**.

Linux implements **pluggable authentication modules (PAMs)** to determine how a user is to be authenticated and whether there are password policies associated with the password databases. PAMs are a series of configuration files that provide dynamic authentication for administrative and other services. The rules associated with PAM files in the /etc/pam.d/ directory can further specify local access limits for different administrative tools and commands.

Some Linux appliances, such as home routers, include default usernames and passwords. It is important to configure such appliances with nondefault usernames and strong passwords. One recently discovered botnet, named for Chuck Norris, can infect and take over such Linux appliances if their administrators retain standard or weak usernames and passwords. The botnet was discovered by Masaryk University researchers in the Czech Republic with the help of a **honeypot**, which tempts black-hat hackers with seemingly valuable data.



```
[kilroy@cozy ~]$ sudo authselect list
- minimal        Local users only for minimal installations
- nis            Enable NIS for system authentication
- sssd           Enable SSSD for system authentication (also for local users onl
y)
- winbind        Enable winbind for system authentication
```

Courtesy of CentOS Project.

FIGURE 2-2 CentOS authselect tool.

Protecting Files with Ownership, Permissions, and Access Controls

Everything on Linux is represented by a file. A directory is a special kind of file, which should be executable for users who are allowed to list files on that directory. If the execute permission isn't set, you won't be able to change into that directory using the `cd` command. Other special kinds of files include devices, commonly used to represent and communicate with certain hardware components; character devices such as sound cards; block devices such as drives and partitions; and soft links, which can connect files across different partitions or volumes.

Every file and directory on a Linux system is owned by a specific user and group. With discretionary access controls, the file or data owner can set different read, write, and executable permissions for the user owner, the group owner, and all other users.

While a lot can be done with read, write, and execute permissions for a file, such permissions may not provide enough control. For an additional layer of permissions, Linux can work with access control lists. To make that happen, the target Linux filesystem(s) must be configured and mounted with the access control list option. Once mounted, you can configure access control masks that supersede regular file permissions.

Current Linux distributions may include one or two levels of administrative access control. To give administrative access to individual utilities to specific users, you would use the `sudo` command and configure the access using the /etc/sudoers file. Later distributions support fine-grained access to hardware components through the **PolicyKit**.

So Many Access Controls

Linux implements access controls in a couple of different ways. Even if you've studied them in other Linux books, it may be difficult to keep them straight. So here's a short primer of various access controls in Linux.

Discretionary access controls as implemented in Linux are read, write, and execute permissions that can be given to users and groups. They are called discretionary because they can be implemented at the discretion of the data owner. Linux includes two types of discretionary access controls. The first is standard read, write, and execute permissions. The second is **access control lists (ACLs)**. ACLs provide a second layer of discretionary access control. Specifically, they support read, write, and execute permissions that may supersede other discretionary access controls.

Mandatory access controls are systems like SELinux and AppArmor. You'll have to choose one or the other, as these systems are not compatible. While SELinux is the default for Red Hat distributions, AppArmor is the default for many other Linux distributions, including Ubuntu.

When used together, discretionary and mandatory access controls provide multiple layers of protection. These controls are usually also coupled with the firewall protection associated with the `iptables` command. Most iptables rules also control access to systems and protocols, making them another layer of access control.

Firewalls and Mandatory Access Controls in a Layered Defense

A firewall may be a common first line of defense when it comes to remotely accessing a system, but it shouldn't be the last. In addition to the firewall, a system administrator may implement TCP Wrappers and **mandatory access controls** using either AppArmor or SELinux. Mandatory access controls are permissions that have been set administratively. Individual users cannot change them.

When it comes to protecting your remote services, however, the starting point should be a firewall. Linux has made several attempts to implement a firewall. The initial implementation, ipfwadm, based on BSD's ipf, was included with Linux 2.0. When Linux 2.2 was released, it included ipchains. That was replaced by iptables in version 2.4. That version of the firewall software for Linux stuck around through version 2.6 and into the 3.0 line. It wasn't until version 3.13 was released in early 2014 that the next iteration arrived. This time, it's called nftables. In addition to nftables, Ubuntu systems include an uncomplicated firewall, which is an interface to iptables. Another firewall option is firewalld, which is managed through the command line option firewall-cmd. No matter which firewall interface you are using, though, iptables is still commonly available on Linux systems and can be used to configure firewall rules.

Firewall Support Options

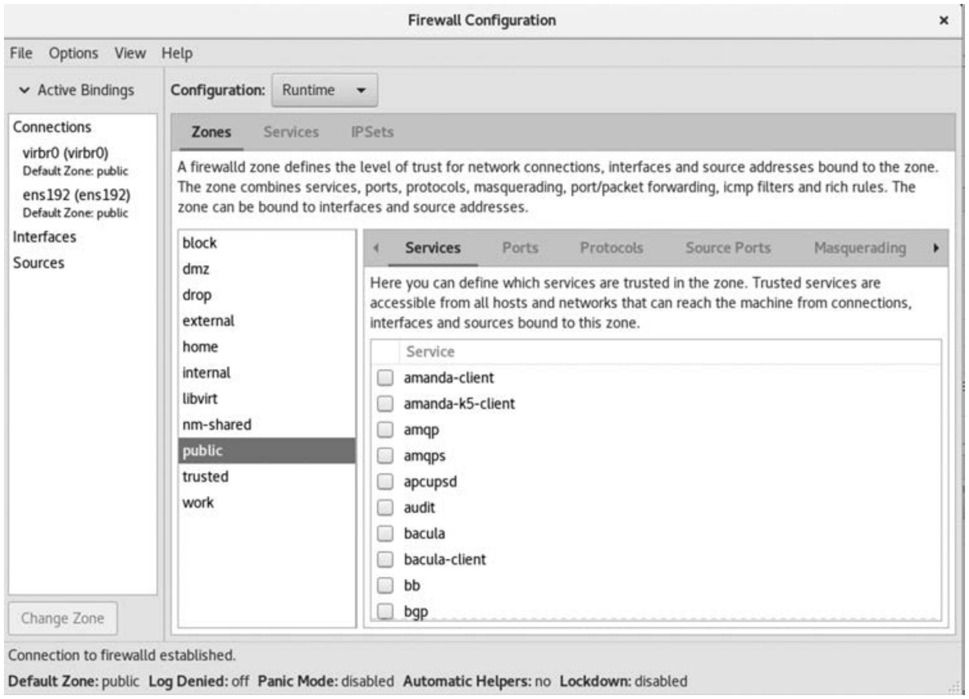
Underneath the hood, no matter what interface you are using, is often iptables. Although you can get a GUI to manage the rules for you, they eventually translate to a set of calls to the iptables program. This program manages the in-memory settings that determine how to categorize and then filter packets flowing through the system. These can be inbound, outbound, or just moving from one interface to another. Iptables can also perform network address translation, where Internet Protocol (IP) addresses are manipulated as a packet passes through the filter. A common set of packet filter rules will look like the following.

```
iptables -A INPUT -s 100.0.0.0 /8 -d 192.168.1.10 -p tcp -m tcp
➥ --dport 80 -j ACCEPT
iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp -j LOG
iptables -A OUTPUT -j ACCEPT
```

In the first rule, the `-A` indicates you are appending to an existing chain named `INPUT`. This chain is used when the filter receives packets coming into the system. It also looks for source and destination addresses. The source address is an address block, while the destination is a host address because it doesn't include the bitmask. After that, you check for the protocol in use by using the `-p` flag. You can also specify a particular matching strategy using the -m flag. In this case, you are just looking to match with `tcp` packets. The parameter `--dport` specifies the destination port, which is the port used for web servers, 80. Finally, with `-j` you indicate where you are jumping. You have the option to `ACCEPT`, `DROP`, or `LOG` as standard jump points. In the third rule, you are performing logging on all output packets that are TCP.

The Linux netfilter modules have always been stateful. The second rule looks at the state of the connection. In this case, you are looking for new connections as well as established ones. The established connections are ones that have previously been allowed through the firewall. You can check for `NEW`, `ESTABLISHED`, or `RELATED`. In the case of `RELATED`, you may be looking at a data flow that is separate from the flows you've seen. One example of this is an FTP transfer. In an active FTP session, the data transfer happens over a different set of ports, which would be related to control session.

Red Hat Enterprise Linux (RHEL) uses a different method for managing firewalls. Where other Linux distributions use iptables, RHEL switched to using firewalld in RHEL7. Firewalld uses zones to categorize rules. You can associate different rules with different zones and use different interfaces with different zones to pick up different rules. This makes it a lot easier for enterprises to more easily support multiple security zones on a system that may have an administrative interface as well as an interface that may be public facing. Another feature of firewalld is specifying where the changes take place. Either they are made to the rules in memory, or they are made on a permanent basis to disk. With iptables, all changes are made in memory. To make them persistent, you had to dump the rules from memory into a file and then read that file in at boot time.



Configuration: Runtime

Default Zone: public (virbr0)

Default Zone: public (ens192)

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

Connection to firewalld established.

Default Zone: public Log Denied: off Panic Mode: disabled Automatic Helpers: no Lockdown: disabled

Courtesy of firewalld.org.

FIGURE 2-3 The firewalld interface.

Figure 2-3 shows the user interface for firewalld. Rather than specifying rules by port numbers, as was the case with iptables, firewalld keeps track of all of the services installed on the system. You specify rules based on the services on the system. This can help to protect the system by not opening up ports for services that are not installed appropriately.

Mandatory Access Control Support

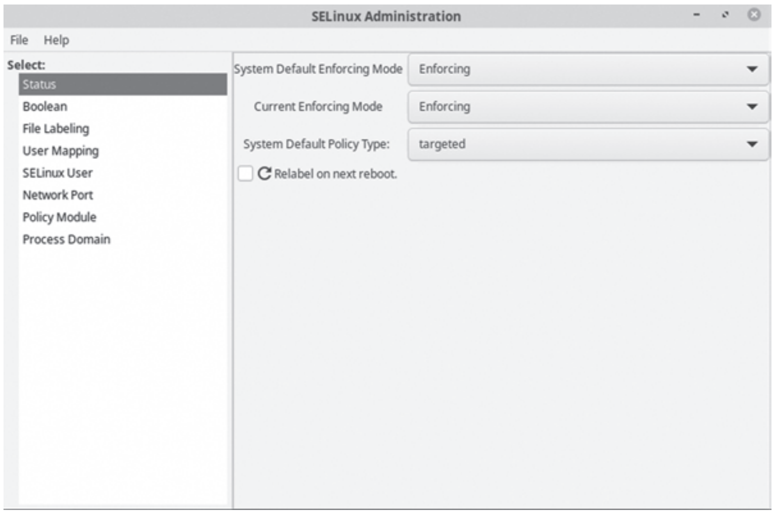
Firewalls have their value, of course, but they are insufficient to protect systems from attack, particularly unexpected access. The mandatory access controls associated with SELinux and AppArmor provide protections at the host level rather than at the network level. When properly configured, these controls can check access to services and commands to make sure they're run only by intended users, in properly labeled directories.

TIP

You can't use both SELinux and AppArmor on the same system.

Proper mandatory access controls make a system more robust. Compromised users, even service users with some level of administrative privileges, can't use other services. However, the configuration of a mandatory access control system requires discipline. For example, you can't just add a new directory to help organize FTP server files. With a mandatory access control system like SELinux, you must make sure those directories are labeled and configured for their intended purposes.

It's normally best to administer a system from the command-line interface. However, some of these tools are tricky. Some GUI tools can help less-familiar administrators understand more about the service. To that end, Red Hat's SELinux Administration tool, shown in **Figure 2-4**, is excellent. On a Red Hat system, this requires the package policycoreutils-gui, which is not installed by default. This screen capture comes from a Fedora Core installation, which is a development distribution for Red Hat. Work that takes place in Fedora Core may eventually turn up in Red Hat Enterprise Linux and, by extension, CentOS and other Red Hat-related distributions.



System Default Enforcing Mode: Enforcing
Current Enforcing Mode: Enforcing
System Default Policy Type: targeted
☐ Relabel on next reboot.

Courtesy of Red Hat, Inc.

FIGURE 2-4

The SELinux administration tool.

In a similar fashion, AppArmor includes profiles for specific commands. You can add profiles for additional commands as you learn more about mandatory access controls. In contrast, SELinux provides extensive policies for different applications as well as users and files. You would create contexts for different system resources and then control what gets access to the different resources.

Both AppArmor and SELinux include a monitoring mode, where violations are logged without preventing access. This can allow you to find failures that may result from the implementation of a mandatory access control package before you implement it. Using the monitor mode, called Permissive in SELinux, for instance, you can make sure you aren't breaking anything. On AppArmor, this is known as complain mode. The resulting log files can help you better customize either of these mandatory access control systems. It also provides ammunition for going back to developers and system administrators to get them to comply with the expectations of a tightened down operating system.

SELinux, originally developed by the National Security Agency, has a reputation for being difficult to manage, especially if you are trying to implement it with specialized services and applications. Understanding how to correctly manipulate the access control lists requires a fairly steep learning curve, but it is possible to have a system that is less prone to infiltration. AppArmor is another approach, using profiles to achieve application protection. AppArmor also attempts to supplement the discretionary access controls provided by Linux with mandatory access controls, just like SELinux, but the implementation is much different.

Protecting Networks Using Encrypted Communication

While network communications is generally being moved by the requirements of users and businesses to being encrypted from end-to-end across the board, we aren't completely there. Some places will still use protocols like FTP for file transmission that may not include encryption for file transfer. As another example, email is commonly sent from end to end in plaintext, though if you are using a webmail interface, the communication between you and the web server is encrypted. However, once the protocol moves from HTTP, which is you to the web server, to SMTP, which is the Simple Mail Transfer Protocol, and the message is being sent from one mail server to another, it's likely the communication is happening without encryption. The difference would be if you are sending a message from one user in the same domain to another, since it wouldn't be sent over the network.

Cleartext communication was common in the early days of Unix, which was used primarily at universities where colleagues were more likely to trust each other. Today's environments are far more hostile, so it's likely you won't run across unencrypted communication, especially in an enterprise environment. It's simply become too easy to implement encrypted communications, like SSH. With the `ssh` command, you can connect to remote systems with the SSH server. The communication—and more importantly, the password—is encrypted by default. You can even set up passphrases to protect the private key used to encrypt communication between the two endpoints. Passphrases are more secure than passwords. Because the passphrase is used solely to give access to the encryption key, it is never transmitted over the network, even in an encrypted manner.

SSH can be used for more than just shell connections. It can also be used to securely transmit backups and more. It can even be used to connect securely to a remote GUI client. While it's possible to tunnel many different connections over SSH, SSH packages commonly come with a secure copy client (scp) that is used to transfer files over an encrypted SSH connection.

FYI

While the Telnet server is not commonly used anymore, the Telnet client is a very useful tool. It can help you verify that a service is accessible. You can use the telnet command both locally and over a network to verify the availability of a service over any TCP/IP port. By using the Telnet client and specifying a port number after the hostname, you can initiate a TCP connection to the port on the host, giving you a raw TCP connection to that port. This is very valuable for manually testing services or quickly verifying that a port is listening.

Tracking the Latest Linux Security Updates

For many regular users, it's good enough to download and install updates provided by a distribution when they become available. But Linux is used by a range of users, from home hobbyists to engineers and scientists who know a lot about Linux, to administrators like you, and more. The response of each of these groups to a security update may vary.

The developers behind most Linux distributions do an excellent job with security updates. They monitor security issues in real time. They monitor the work of the open-source community as security issues are addressed.

Linux Security Updates for Regular Users

Generally, the package updates released for a distribution have been tested with standard configurations. Linux users who don't have a lot of time to maintain their systems may choose to set up automatic updates. That works if the user trusts the company and the open-source community behind a distribution. Alternatively, you can read every security bulletin to see if and how the problem affects the systems on your networks.

In general, it's not too difficult to back out of a security update. In the worst case, configuration files can be saved. Updates can be uninstalled. Older versions of updated packages can be installed again.

Kernel updates may be kept out of automatic updates to ensure the stability of the system. If you do update the kernel, the update package will leave the existing kernel in place and configured in the boot loader. This ensures that you have a fallback option in case the new kernel doesn't work as expected, either as it is or with applications you are running.

Linux Security Updates for Home Hobbyists

While this perception is changing a lot because of the current proliferation of desktop-oriented distributions, people may think of Linux users as tinkerers—people who like to experiment with their computers. Home hobbyists may like to work with the latest in hardware. Some hardware manufacturers develop drivers for Microsoft operating systems first. Even then, not all hardware manufacturers release their drivers under an open-source license. Linux developers frequently have to compile their own code to work with the latest hardware. The first versions of compiled code may be tied to a specific kernel release.

Some specialty software may also be tied to a specific version of a kernel. Examples include some VMware and Virtualbox virtual machine systems. If you install a new kernel, you'll need to reinstall or recompile the code for that software. If you install a kernel security update, you're installing a different kernel release. Kernel drivers, like those used by virtualization software, must be rebuilt to work with the new kernel. In many cases, the kernel modules are tagged with the kernel version and won't work with any other version. Most importantly, though, the modules have to be built to ensure all the right kernel pieces are in place to support the driver.

Linux Security Updates for Power Users

While Linux distributions do a good job of keeping software up-to-date and also maintaining a large number of packages available for installation, there may still be software packages some users want to install. It's possible some users will also want to custom compile a software package without relying on the choices made by the package maintainers with the distribution. Compiling software is a slightly slower process than installing a package, and it can also result in errors if it's not configured correctly with the correct libraries. However, depending on the level of control the user wants, that user may expect to self-compile their own software. This would mean they would have to keep the software up-to-date themselves with patches and updates. Some so-called power users may be up to this challenge and even relish it.

Other types of users have other needs. While you generally want to limit the number of packages installed on a system and reduce the amount of administrative control any user has, this isn't always possible. Developers, for example, need ready access to administrative tools such as compilers. Compilers can be used by malicious users or attackers to build custom software that can be employed to launch further attacks. This concern was far more valid when there was less homogeneity across Unix implementations. If a malicious user gets into one Linux system, the same pre-built binary will work on that Linux system just as it will work on any other Linux system. This is because the application binary interface that all programs use is the same across versions and implementations of Linux, or else it's no longer Linux. The only variation here would be if you ran across a very, very old version of Linux. In that case, the application binary interface might be different enough that a newer binary may not work. The same is true if you try to move between 32-bit and 64-bit systems with applications.

Security Updates for Linux Administrators

As a Linux security professional, you should read all security bulletins related to software on your systems. You should consider the effects of updates before deciding whether they should be installed.

If you determine that the security bulletin does not apply to the systems on your network, don't install that update. Unless some changes are made, that update will still be there. Depending on several factors, the updated software may be in a distribution-specific repository or a download from a public server maintained by the developers of the software in question.

To deviate from updates provided for a distribution, you can create a local update repository, loaded with packages customized for local systems. You then configure the systems on the local network to take their updates from that local repository. Yes, that may involve some extra additional work, but a local update repository can also save bandwidth on a network's connection to the Internet.

In other words, every Linux workstation and server normally gets its updates from remote repositories somewhere on the Internet. All packages downloaded for a specific distribution are the same. Linux workstations may require periodic updates of huge packages, such as for the OpenOffice.org suite. If you create a local update repository, these packages are downloaded from the Internet only once. The updates can then be pulled over the local network from each system.

Linux Security Update Administration

If you administer just a few Linux systems, you can manage their updates directly. As the administrator, you may be able to administer those systems remotely with a tool such as SSH. You can then run needed update commands on those systems.

If you administer a substantial number of Linux systems, consider the centralized management tools available for several Linux distributions. Tools such as the Red Hat Network, Canonical's Landscape, and Novell's Zenworks support the administration of groups of computers from a centralized interface. While these tools require subscription fees, they do save time. If you manage that many systems, you are probably already paying one of these companies for a support services anyway (and if you aren't, you probably should be).

With these tools, you can set up a single command or a script to be run on a group of computers. For example, you might set up different update commands for servers, engineering workstations, and data-entry systems.

The Effect of Virtualization on Security

Virtualization makes it easy to start with a very secure baseline system. Because virtual-machine hard disks reside on only a few very large files, it takes just a few commands to create new virtual machines. In other words, you can start with a stripped-down, well-hardened system and add a single service, blocking packets to all unused ports, and securing every other account and directory with appropriate mandatory access controls.

Virtualization is essential to most computing environments, whether it's a small company or a much larger enterprise network. One of the issues with virtualization is that you don't have physical systems to look at. Instead, you have a lot of logical systems that may be much easier to overlook. A virtual machine that is forgotten is not updated. And a system that is not updated is an open invitation to a malicious user to break into your system. If that system remains forgotten, it can serve as a foothold—a way for that malicious user to break into more important parts of your network.

While virtualization has roots in the time sharing found on mainframe systems, it has taken a number of new turns for modern computers. There are five different categories of virtualization:

- **Applications**—While a number of companies are developing virtual machines that encapsulate single applications, Linux already has an application-level virtualization tool known as Wine Is Not an Emulator (WINE). A more common approach to application-level virtualization today is containerization with packages like either Linux Containers (LXC) or Docker. This type of virtualization restricts the memory used by an application by tagging it. If the operating system detects the application trying to access memory that isn't similarly tagged, it won't be allowed. This prevents one application from corrupting the memory space of another application and an attacker jumping from one process to another.
- **Platform-level virtual machines**—The first PC-based virtual machines were applications installed in operating systems. Examples of virtual machines installed as applications include VMware Player, VMware Workstation, Virtualbox (open-source edition), and Parallels Desktop. In the language of virtualization, these are called Type 2 hypervisors. A **hypervisor** is the software that manages and runs the virtual machines. In these cases, there is a host operating system that sits between the machine (or bare metal) and the hypervisor, or virtualization software.
- **Paravirtualization**—A software interface for virtual machines with limited resources. Paravirtualized virtual machines can be configured on older CPUs.
- **Hardware-assisted virtualization**—The processor here has extensions to support various supervisory functions in hardware rather than having to support something like an emulator in software. For the last several years, both AMD and Intel have supported these virtualization extensions in their CPUs.
- **Bare metal virtualization**—In the language of virtualization, these are called Type 1 hypervisors, and the virtualization software runs directly on the hardware without a host operating system. VMWare's ESX Server and Citrix's Xen Server as well as Microsoft's Virtual Server are all examples of Type 1 hypervisors that operate on the bare metal.

To confirm that a system can handle hardware virtualization, examine the /proc/cpuinfo file. If it is an Intel CPU, it should include the `vmx` flag. If it is a CPU from Advanced Micro Devices (AMD), it should include the `svm` flag.

Variations Between Distributions

One of the hallmarks of open-source software is diversity. For example, you can select from at least four different SMTP servers to manage email on a network. While some distributions focus on system administration, others focus on consumer applications such as multimedia. There are even live CD Linux distributions with security tools.

Despite the diversity in standard services, you can generally install (or remove) the software, services, and applications of your choice on any Linux system. For example, while Ubuntu developers include **Postfix** as the default SMTP server, you can install other SMTP servers, including sendmail, **Exim**, and **Qmail**, directly from Ubuntu repositories.

A Basic Comparison: Red Hat and Ubuntu

This book will spend some time focusing on two Linux distributions and their related child distributions: Red Hat Enterprise Linux and Ubuntu Server Edition. The developers behind these distributions have made different choices. Although the Red Hat and Ubuntu distributions occupy a large share of the Linux marketplace, there may be another distribution that better fits your needs. Understanding the differences between these distributions can help you make a better choice.

The biggest difference may be the package-management system. Not surprisingly, Red Hat uses the Red Hat Package Manager (RPM) to build packages from source code and install them on the Red Hat distribution. Red Hat also uses a utility called the Yellowdog Update, Modified (yum) to actually install the RPM packages to the system while also checking for dependencies. In recent versions, it has been replaced by Dandified yum (dnf). Because Ubuntu built its distribution on top of the foundation of Debian Linux, it uses the Debian Package Manager to build and install its packages. This is commonly done with the command line utility apt.

There are also similarities between these distributions. For example, Red Hat and Canonical (Ubuntu) both release distributions with long-term support. Red Hat supports its Enterprise Linux releases for seven years. The most recent version of Red Hat Enterprise Linux (RHEL) is RHEL8, with updates to that version being released regularly.

Ubuntu releases distributions every six months and releases more robust distributions every two years. These releases are noted as long-term support (LTS) releases. The server edition of every Ubuntu LTS release is supported for five years.

Along with long-term support, Red Hat and Canonical have made a number of other similar decisions. Both companies offer subscriptions to a web-based support tool that allows authorized users to administer groups of Linux systems remotely.

Both Red Hat and Canonical have made similar choices for some default applications. Web servers, mail servers, and other Internet-facing services are available in different packages. If you wanted to install an application, for instance, you could use Apache or Nginx. If you needed a mail server, you could use Postfix or Exim. Incidentally, both Red Hat and Canonical use the GNU Network Object Model Environment (GNOME) as the default graphical desktop, although the actual implementation looks different from one system to another because of custom modifications.

There are differences that go beyond the package-management system, however. For example, Ubuntu includes a minimal installation option with its server release, well suited for bastions on virtual machines. That installation includes a kernel optimized to work as part of a virtual guest.

Red Hat and Ubuntu have also made different choices in terms of default applications. Of course, you can override these with the applications of your choice. In many cases, Red Hat and Ubuntu also support these alternatives.

Sometimes, Red Hat and Ubuntu learn from each other. Given their releases under open-source licenses, they can take the source code from a competitive distribution and use it themselves. Ubuntu uses a number of GUI administrative tools originally built for Red Hat.

More Diversity in Services

Linux includes a diverse range of services. If a security issue arises with one service, it's normally best to install the update or patch to address the security issue. Sometimes, however, that update might cause more severe problems.

In that case, you may want to consider a different service. **Table 2-3** lists a number of alternatives for different services. It is not by any means a complete list. However, it does describe alternatives if one service option has a fatal security flaw. It also gives you an idea of how many services could be installed on a single server.

TABLE 2-3 Alternatives for different services.

FUNCTION	SERVICE OPTIONS	NOTES
Domain Name System (DNS)	**Berkeley Internet Name Domain (BIND)** **Daniel J. Bernstein's DNS (djbdns)**	DNS is a hierarchical database of domain names and IP addresses. It enables you to look up hostnames from IP addresses and vice versa. BIND and djbdns are both implementations of a DNS server. BIND is the most common DNS server on the Internet. Djbdns is a relatively lightweight DNS server alternative to BIND.
File transfer	Professional FTP Daemon (ProFTPD) Troll FTP vsftp **Trivial File Transfer Protocol (TFTP)**	TFTP is a protocol and service that uses a simplified form of FTP. It is a very lightweight means of transferring files with very few commands available.
Graphical desktop environment	GNOME K Desktop Environment (KDE) Xfce Lightweight X11 Desktop Environment (LXDE) Cinnamon Mate	In Linux, the graphical desktop environment is separate from but requires the use of an X Windows System Server. It may also include a window manager to control the placement of windows within that GUI. Both Cinnamon and Mate are implementations of an older version of GNOME, prior to the Unity interface.
Graphical user interface	X.org **XFree86** Freedesktop.org	XFree86 is an implementation of the X graphical interface.

(Continued)

TABLE 2-3 Alternatives for different services. *(Continued)*

FUNCTION	SERVICE OPTIONS	NOTES
Graphical login manager	**GNOME Display Manager (GDM)** **KDE Display Manager (KDM)** **X Display Manager (XDM)** **LightDM**	A graphical login manager is a service for graphical logins to a Linux GUI. GDM is a graphical login manager built by the developers of the GNOME Desktop Environment. KDM is a graphical login manager built by the developers of KDE. XDM is a graphical login manager built by the developers of X.Org GUI server.
Mail user agents	**Cyrus** **Dovecot**	Cyrus is an email server developed at Carnegie-Mellon University, primarily for IMAP version 4 email delivery. Dovecot is an open-source email service designed for regular and secure versions of the POP and IMAP protocols. Cyrus and Dovecot are implementations of mail servers that are used to retrieve mail for clients.
Printing	CUPS System V **Line Printer next generation (LPRng)**	LPRng is a server used to send print jobs to or from users or systems.
Remote connections	SSH Remote Shell (RSH) Telnet **Kerberos Telnet**	Kerberos Telnet is a version of the Telnet server that can use Kerberos tickets to enhance security.
Mail transport agents	Postfix sendmail Sendmail Exim Qmail	
Structured Query Language (SQL) databases	**MySQL** **MariaDB** **PostgreSQL** Proprietary options from Sybase Oracle	MySQL is open-source database program, currently owned by Oracle. PostgreSQL is an open-source database program sponsored by a variety of open-source and other IT companies.

(Continued)

TABLE 2-3 Alternatives for different services. *(Continued)*

FUNCTION	SERVICE OPTIONS	NOTES
Web server	Apache Boa Caudium Lighthttpd Nginx Roxen Sun Java Zeus	

CHAPTER SUMMARY

Linux security starts with the kernel. The modularity of the Linux kernel means that it is more robust. Many modular options affect security. To that end, you need to choose whether to accept the kernel as customized for your distribution or whether to customize and compile the kernel yourself. Linux security continues with the boot process. Good security prevents unauthorized users from booting from a live CD and selecting administrative options from the GRUB boot loader. Beyond the basic operating system, you should keep active services to a minimum. If possible, don't install a GUI.

File ownership and permissions start with discretionary access controls, enhanced by access control lists. The use of sudo can provide restricted administrative privileges and eliminate the need for someone to be logged in as root, which can be dangerous. Host-based firewalls using iptables or firewalld will help to prevent unwanted network communications from getting into the system. You can provide additional protection on services that have to be exposed through the firewall by using TCP Wrappers through the internet super server or by using additional restrictions built into specific services. Mandatory access control systems such as SELinux and AppArmor add another layer of control, protecting your system even if there is a security breach. With the availability of SSH, there is no reason to use cleartext protocols such as Telnet.

With respect to security, you may need to pick and choose whether to download a security update. Some security updates may not apply to all configurations, although automatic update services will determine whether distribution-provided updates are relevant to your system. If they are, they will either be updated automatically or be presented to you as options for updates. Virtualization adds another dimension to security. Isolating services to appropriate security zones within the same physical host will help to improve the security of other systems. Services or systems that need different levels of protection can be put on other virtual machines, limiting points of attack. Finally, if you understand the differences between Linux distributions, you can make better decisions when it comes to choosing security options for the network.

KEY CONCEPTS AND TERMS

Access control lists (ACLs)
Berkeley Internet Name Domain (BIND)
Binary kernel
CentOS
Common Unix Printing System (CUPS)
Cyrus
Daniel J. Bernstein's DNS (djbdns)
Default runlevel
Discretionary access controls
Domain Name System (DNS)
Dovecot
Exim
File Transfer Protocol (FTP)
GNOME Display Manager (GDM)
Grand Unified Bootloader (GRUB)
Graphical desktop environment
Graphical login manager
Honeypot
Hypervisor
iptables
KDE Display Manager (KDM)
Kerberos Telnet
Knoppix
LightDM
Line Printer next generation (LPRng)
Linux distribution
Linux kernel
Linux Kernel Organization
Live CD
Mandatory access controls
MariaDB
Modular kernel
Monolithic kernel
MySQL
Network Time Protocol (NTP)
Patch
Pluggable authentication modules (PAMs)
PolicyKit
Postfix
PostgreSQL
Qmail
Runlevel
sendmail
Sendmail
Source code
sshd
syslog
Trivial File Transfer Protocol (TFTP)
Very Secure File Transfer Protocol (vsftpd)
Winbind
X Display Manager (XDM)
XFree86

CHAPTER 2 ASSESSMENT

1. Which of the following statements best describes the structure of the Linux kernel?

A. A single monolithic kernel
B. A completely modular kernel
C. A modular core with monolithic components
D. A monolithic core with modular components

2. The website associated with the Linux Kernel Organization is ________.

3. Which of the following statements is *not* true about a live CD distribution? Assume your system can boot from appropriate locations.

A. It can be booted from a DVD drive.
B. It can be booted from a USB port.
C. It automatically installs that Linux distribution on your system.
D. It provides administrative control of your system without a password.

4. Which of the following services should *not* be disabled on a bastion host used as an FTP server? Assume that the host is administered remotely, over an encrypted connection.

A. SSH
B. Telnet
C. CUPS
D. iptables

5. Which of the following is *not* a potential security issue with respect to the Linux GUI?

A. The Linux GUI is a client-server system.
B. Linux GUI applications can be networked.
C. Linux GUI applications can be accessed over an SSH connection.
D. Users can log into the Linux GUI remotely.

6. Which of the following authentication tools work locally?

A. NIS
B. PAM
C. LDAP
D. Winbind

7. Which of the following is an example of discretionary access controls?

A. SELinux
B. AppArmor
C. PolicyKit
D. User-defined read, write, and execute permissions

8. Which of the following options is *not* used to block access from certain IP addresses?

A. iptables
B. SELinux
C. TCP Wrappers
D. Extended internet super server

9. Which of the following statements best describes the role of mandatory access controls?

A. They protect other services after a security breach in an account.
B. They protect a system from access by a malicious user through firewalls.
C. They disable cleartext services such as Telnet.
D. They provide specific requirements for access to critical services.

10. Packages associated with SSH include a client for which of the following protocols?

A. Samba
B. FTP
C. Telnet
D. SMTP

11. What is the best course of action if you want to take control of those packages that are updated on your distribution?

A. Create your own update repository.
B. Deselect the packages that should not be updated.
C. Change to a different distribution.
D. Use the update repositories from a different distribution.

12. Which of the following is *not* a standard open-source option for SMTP email services?

A. sendmail
B. Postfix
C. Dovecot
D. Exim