4

RHCSA-Level Security Options

CERTIFICATION OBJECTIVES

4.01 Basic File Permissions

4.02 Access Control Lists and More

4.03 Basic Firewall Control

4.04 A Security-Enhanced Linux Primer

✓ Two-Minute Drill

Q&A Self Test

Linux security starts with a concept known as "discretionary access controls." They include the permissions and ownership associated with files and directories. Default permissions on new files depend on the umask. Permissions can go further with specialized bits. Linux discretionary access controls can be configured on a more fine-grained basis with the help of access control lists (ACLs). Those ACLs support permissions given to specific users, overriding standard ownership and permissions.

Also in the realm of security is the firewall. In this chapter, you'll examine the default firewall, how it works with the **iptables** command, and how it can also be configured with Red Hat firewall configuration tools. What you create can be further protected with a different kind of security known as "mandatory access control." The RHEL 6 implementation of such is known as Security-Enhanced Linux (SELinux). Red Hat expects you to work with SELinux enabled on their exams. To that end, you'll examine how to set enforcing modes, change file contexts, use boolean settings, and diagnose SELinux policy violations.

If you're starting with the default installation created during the installation process, you may need to install additional packages during this chapter. If a network installation is available, take the name of the package and apply the **yum install** command to it. For example, to review the GUI-based firewall configuration tool, you'll need to install it with the following command:

```
# yum install system-config-firewall
```

For more information on the process, see Chapter 7.

INSIDE THE EXAM

Basic File Permissions

Security in Linux starts with the permissions given to files. As everything in Linux can be defined as a file, it's an excellent start. In any case, the related objective, once understood, is fairly straightforward:

- List, set, and change standard ugo/rwx permissions

Standard permissions for Linux files are defined for users, groups, and others, which leads to the *ugo*. Those permissions are read, write, and execute, which defines the *rwx*.

Such permissions are defined as discretionary access control, to contrast with the mandatory access control system known as SELinux, also discussed in this chapter.

Access Control Lists

ACLs can be configured to override basic file permissions. For example, with ACLs, you can set up a file in your home directory that can be read by a limited number of other users and groups. The related RHCSA objective is:

- Create and manage Access Control Lists (ACLs)

Firewall Control

As configured in Linux, a firewall can block traffic on all but a few network ports. It also can be used to regulate traffic in a number of other ways, but that is the province of the RHCE exam. The related RHCSA objective is:

- Configure firewall settings using system-config-firewall or iptables

Security-Enhanced Linux

There's no way around it. On the Red Hat exams, you're expected to work with SELinux. It's not clear whether you can even pass the Red Hat exams unless at least some services are configured with SELinux in mind. To help exam candidates understand what's needed, Red Hat has broken down SELinux-related objectives. The first objective is fundamental to SELinux, as it relates to the three modes available for SELinux on a system (enforcing/permissive/disabled):

- Set enforcing/permissive modes for SELinux

The next objective requires that you understand the SELinux contexts defined for different files and processes. While the associated commands are straightforward, the available contexts are as broad as the number of services available on Linux:

- List and identify SELinux file and process contexts

As you experiment with different SELinux contexts, mistakes happen. You may not remember the default contexts associated with important directories. But with the right commands, you don't have to remember everything, as suggested by the following objective, it's relatively easy to restore the default:

- Restore default file contexts

Finally, the last objective may seem complex. But the boolean settings associated with SELinux have descriptive names. Excellent tools are available to further clarify those boolean contexts that are available. In essence, this means to run a certain service under SELinux, all you need to do is turn on one (or more) switches:

- Use boolean settings to modify system SELinux settings

CERTIFICATION OBJECTIVE 4.01

Basic File Permissions

The basic security of a Linux computer is based on file permissions. Default file permissions are set through the **umask** shell variable. Super user permissions can be configured to give all users and/or groups access to specific files. These are known as the super user ID (SUID) and super group ID (SGID) special permission bits. Ownership is based on the default user and group IDs of the person who created a file. The management of permissions and ownership involves commands such as **chmod**, **chown**, and **chgrp**. Before exploring these commands, it's important to understand the permissions and ownership associated with a file.

File Permissions<MBI> and Ownership

Linux file permissions and ownership are straightforward. As suggested by the related objective, they're read, write, and execute, classified by the user, the group, and all other users. Consider the following output from **ls -l /sbin/fdisk**:

```
-rwxr-xr-x.  1 root  root 103432 Aug 13 01:23  /sbin/fdisk
```

The permissions are shown on the left side of the listing. Ten characters are shown. The first character determines whether it's a regular or a special file. The remaining nine characters are grouped in threes, applicable to the file owner (user), the group owner, and everyone else on that Linux system. The letters are straightforward: *r* = read, *w* = write, *x* = execute. These permissions are described in Table 4-1.

TABLE 4-1

Description of File Permissions

Position	Description
1	Type of file; - = regular file, *d* = directory, *b* = device, *l* = linked file
234	Permissions granted to the owner of the file
567	Permissions granted to the group owner of the file
890	Permissions granted to all other users on the Linux system

It's common for the user and group owners of a file to have the same name. In this case, the root user is a member of the root group. But they don't have to have the same name. For example, directories designed for collaboration between users may be owned by a special group. As discussed in Chapter 8, that involves groups with several regular users as members.

There's a relatively new element with permissions. It's subtle. Notice the dot after the last x in the output to the ls -l /sbin/fdisk command? It specifies control by SELinux. If you've configured ACL permissions on a file, that dot is replaced by a plus sign (+). But that symbol doesn't override SELinux control.

You need to consider another type of permission: the special bit. Not only are these the SUID and SGID bits, but also another special permission known as the sticky bit. An example of the SUID bit is associated with the **passwd** command, in the /usr/bin directory. The **ls -l** command on that file leads to the following output:

```
-rwsr-xr-x. 1 root root 31768 Jan 28 2010 /usr/bin/passwd
```

The **s** in the execute bit for the user owner of the file is the SUID bit. It means the file can be executed by other users with the authority of the file owner, the root administrative user. But that doesn't mean that any user can change other user's passwords. Access to the **passwd** command is further regulated by Pluggable Authentication Modules (PAM) described in Chapter 10, an RHCE skill.

An example of the SGID bit can be found with the **ssh-agent** command, also in the /usr/bin directory. It has the SGID bit to properly store passphrases discussed in Chapter 11, which supports the most secure connections between SSH client and server. The **ls -l** command on that file displays the following output:

```
-rwxr-sr-x. 1 root nobody 112000 Aug 12 07:04 /usr/bin/ssh-agent
```

The **s** in the execute bit for the group owner of the file (group nobody) is the SGID bit.

Finally, an example of the sticky bit can be found in the permissions of the /tmp directory. It means users can copy their files to that directory, while retaining ownership of those files (which is the "sticky"). The **ls -ld** command on that directory shows the following output:

```
drwxrwxrwt. 22 root root 4096 Dec 15 17:15 /tmp/
```

The **t** in the execute bit for other users is the sticky bit.

The Loophole in Write Permissions

It's easy to remove write permissions from a file. For example, if you wanted to make the license.txt file "read-only", the following command removes write permissions from that file:

```
$ chmod a-w license.txt
```

But the user that owns the file can still make changes. It won't work in GUI text editors such as gedit. It won't even work in the nano text editor. However, if a change is made in the vi text editor, the user who owns that file can override a lack of write permissions with the bang character, which looks like an exclamation point (!). In other words, while in the vi editor, the user who owns the file can run the following command to override the lack of write permissions:

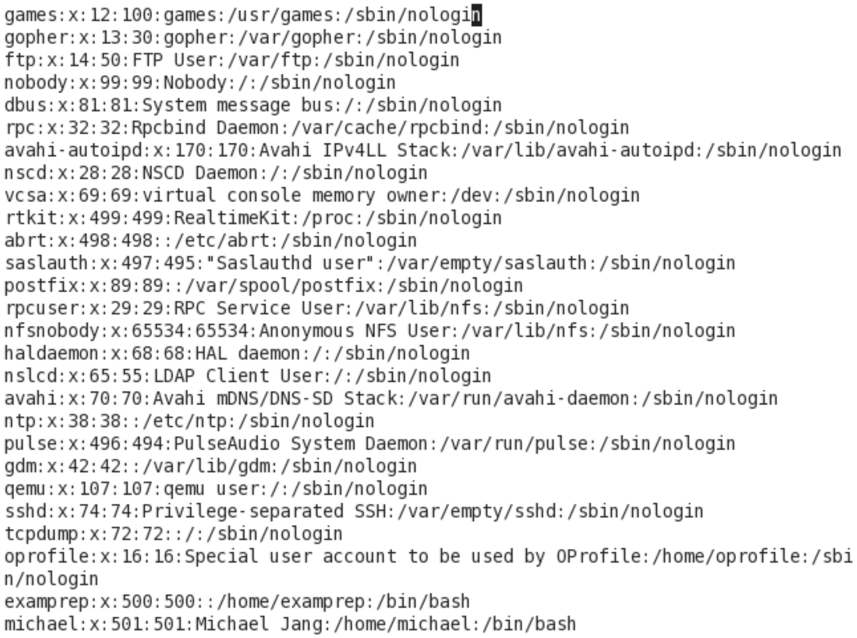
```
!w
```

Basic User and Group Concepts

Linux, like Unix, is configured with users and groups. Everyone who uses Linux is set up with a username, even if it's just "guest." There's even a standard user named "nobody". Take a look at /etc/passwd. One version of this file is shown in Figure 4-1.

FIGURE 4-1

The /etc/passwd file.



```
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-autoipd:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rtkit:x:499:499:RealtimeKit:/proc:/sbin/nologin
abrt:x:498:498::/etc/abrt:/sbin/nologin
saslauth:x:497:495:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
qemu:x:107:107:qemu user:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbi
n/nologin
examprep:x:500:500::/home/examprep:/bin/bash
michael:x:501:501:Michael Jang:/home/michael:/bin/bash
```

As shown, all kinds of usernames are listed in the /etc/passwd file. Even a number of Linux services such as mail, news, ftp, and apache have their own usernames. In any case, the /etc/passwd file follows a specific format, described in more detail in Chapter 8. For now, note that the only regular users shown in this file are examprep and michael, their user IDs (UID) and group IDs (GID) are 500 and 501, and their home directories match their usernames. The next user gets UID and GID 502, and so on.

This matching of UIDs and GIDs is based on the Red Hat user private group scheme. Now run the **ls -l /home** command. The output should be similar to the following.

```
drwx------. 4 examprep examprep  4096 Dec 15 16:12 examprep
drwx------. 4 michael michael  4096 Dec 16 14:00 michael
```

Pay attention to the permissions. Based on the rwx/ugo concepts described earlier in this chapter, only the named user owner has access to the files in their home directories.

The umask

The way **umask** works in Red Hat Enterprise Linux may be surprising, especially if you're coming from a different Unix-style environment. You cannot configure **umask** to allow the automatic creation of new files automatically with executable permissions. This promotes security: if fewer files have executable permissions, fewer files are available for a cracker to use to run programs to break through your system.

on the job

In the world of Linux, a* hacker *is a good person who simply wants to create better software. A* cracker *is someone who wants to break into your system for malicious purposes. In the world of computer security, these terms may be translated to "white-hat hacker" and "black-hat hacker," respectively.

Every time you create a new file, the default permissions are based on the value of **umask**. In the past, the value of **umask** canceled out the value of numeric permissions on a file. For example, if the value of **umask** is 000, the default permissions for any file created by that user were once 777 – 000 = 777, which corresponds to read, write, and execute permissions for all users. They're now 666, as regular new files can no longer get executable permissions. Directories, on the other hand, require executable permissions so that any file contained therein can be read.

When you type the **umask** command, the command returns a four-number output such as 0245. As of this writing, the first number in the **umask** output is always 0 and is not used. In the future, this first number may be usable to allow for new files that automatically include the SUID or SGID bits.

Also, no matter what the value of **umask**, new files in Red Hat Enterprise Linux can no longer be automatically created with executable permissions. In other words, a **umask** value of 0454 leads to identical permissions on new files as a **umask** value of 0545. You need to use commands such as **chmod** to set executable permissions on a specific file.

The Default umask

With that in mind, the default **umask** is driven by the /etc/bashrc file, specifically the following stanza, which drives a value for **umask** depending on the value of the UID:

```
if [ $UID -gt 199 ] && [ "'id -gn'" = "'id -un'" ]; then
    umask 002
else
    umask 022
fi
```

In other words, the **umask** for user accounts with UIDs of 200 and above is 002. In contrast, the **umask** for UIDs below 200 is 022. In RHEL 6, service users such as adm, postfix, and apache have lower UIDs; this affects primarily the permissions of the log files created for such services. Of course, the root administrative user has the lowest UID of 0. By default, files created for such users have 644 permissions; directories created for such users have 755 permissions.

In contrast, regular users have a UID of 500 and above. Files created by such users normally have 664 permissions. Directories created by such users normally have 775 permissions.

Commands to Change Permissions and Ownership

Key commands that can help you manage the permissions and ownership of a file are **chmod**, **chown**, and **chgrp**. In the following subsections, you'll examine how to use those commands to change permissions along with the user and group that owns a specific file, or even a series of files.

One tip that can help you change the permissions on a series of files is the **-R** switch. It is the recursive switch for all three of these commands. In other words, if you specify the **-R** switch with any of the noted commands on a directory, it applies the changes recursively. The changes are applied to all files in that directory, including all subdirectories. Recursion means that the changes are also applied to files in each subdirectory, and so on.

The chmod Command

The **chmod** command uses the numeric value of permissions associated with the owner, group, and others. In Linux, permissions are assigned the following numeric values: *r* = 4, *w* = 2, and *x* = 1. For example, if you were crazy enough to want to give read, write, and execute permissions on **fdisk** to all users, you could run the **chmod 777 /sbin/fdisk** command. The **chown** and **chgrp** commands adjust the user and group owners associated with the cited file.

The **chmod** command is flexible. You don't always have to use numbers. For example, the following command sets execute permissions for the user owner of the Ch3Lab1 file:

```
# chmod u+x Ch3Lab1
```

Note how the **u** and the **x** follow the ugo/rwx format specified in the associated RHCSA objective. To interpret, this command adds (with the plus sign) for the user owner of the file (with the **u**) execute permissions (with the **x**).

These symbols can be combined. For example, the following command disables write permissions for the group owner and all other users on the local file named special:

```
# chmod go-w special
```

While you can use all three user types in the **chmod** command, it's not necessary. As described in the labs in Chapter 3, the following command makes the noted file executable by all users:

```
# chmod +x Ch3Lab2
```

For the SUID, SGID, and sticky bits, some special options are available. If you choose to use numeric bits, those special bits are assigned numeric values as well, where SUID=4, SGID=2, and sticky bit=1. For example, the following command configures the SUID bit. It includes rwx permissions for the user owner, rx permission for the group owner, and r permissions for other users, on the file named testfile:

```
# chmod 4764 testfile
```

If you'd rather use the ugo/rwx format, the following command activates the SGID bit for the local testscript file:

```
# chmod g+s testscript
```

And the following command turns on the sticky bit for the /test directory:

```
# chmod o+t /test
```

While the **chmod** command described in this section assumes changes are made by the root administrative user, that's not always required. The user owner of a file is allowed to change the permissions associated with that file.

The chown Command

The **chown** command can be used to modify the user that owns a file. For example, take a look at the ownership for the first figure that I created for this chapter, based on the **ls -l** command:

```
-rw-r--r--. 1 michael examprep 855502 Oct 25 14:07 F04-01.tif
```

The user owner of this file is michael; the group owner of this file is examprep. The **chown** command shown changes the user owner to user elizabeth:

```
# chown elizabeth F04-01.tif
```

You can do more with **chown**; for example, the following command changes both the user and group owner of the noted file to user donna and group supervisors, assuming that user and group already exists.

```
# chown donna.supervisors F04-01.tif
```

The chgrp Command

You can change the group owner of a file with the **chgrp** command. For example, the following command changes the group owner of the noted F04-01.tif directory to the group named project (assuming it exists):

```
# chgrp project F04-01.tif
```

Special File Attributes

Just beyond regular rwx/ugo permissions are file attributes. Such attributes can help you control what anyone can do with different files. While the **lsattr** command lists current file attributes, the **chattr** command can help you change those attributes. For example, the following command protects /etc/fstab from accidental deletion, even by the root administrative user:

```
# chattr +i /etc/fstab
```

With that attribute, if you try to delete that file as the root administrative user, you'll get the following response:

```
# rm /etc/fstab
rm: remove regular file `/etc/fstab'? y
rm: cannot remove `/etc/fstab': Operation not permitted
```

The **lsattr** command shows how the previous **chattr +i** command added the immutable attribute to /etc/fstab:

```
# lsattr /etc/fstab
----i--------e- /etc/fstab
```

Of course, the root administrative user can unset that attribute with the following command. Nevertheless, the initial refusal to delete the file should at least give pause to that administrator before changes are made:

```
# chattr -i /etc/fstab
```

Several key attributes are described in Table 4-2. Other attributes, such as **c** (compressed), **s** (secure deletion), and **u** (undeletable) don't work for files stored in the ext2, ext3, and ext4 filesystems. The extent format attribute is associated with ext4 systems.

TABLE 4-2

File Attributes

Attribute	Description
append only **(a)**	**Prevents deletion, but allows appending to a file—for example, if you've run chattr +a tester, cat /etc/fstab >\> tester** would add the contents of /etc/fstab to the end of the tester file.
no dump **(d)**	**Disallows backups of the configured file with the dump** command.
extent format **(e)**	**Set with the ext4 filesystem; an attribute that can't be removed.**
immutable **(i)**	**Prevents deletion or any other kind of change to a file.**
indexed **(I)**	**Set on directories for indexing with hashed trees; an attribute that can't be removed.**

CERTIFICATION OBJECTIVE 4.02

Access Control Lists and More

There was a time where users had read access to the files of all other users. But by default, users have permissions only in their own directories. With ACLs, you can give selected users read, write, and execute permissions to selected files in your home directory. It provides a second level of discretionary access control, a method that supports overriding of standard ugo/rwx permissions.

Strictly speaking, regular ugo/rwx permissions are the first level of discretionary access control. In other words, ACLs start with the ownership and permissions described earlier in this chapter. You'll see how that's displayed with ACL commands shortly.

To configure ACLs, you'll need to configure the appropriate filesystem with the **acl** option. Next, you'll need to set up execute permissions on the associated directories. Only then can you configure ACLs with desired permissions for appropriate users.

Now that RHEL 6 uses the Network File System (NFS) version 4, these ACLs can be shared over a network.

Every File Already Has an ACL

As suggested by the title, every file already is configured with an access control list. Assuming the acl package is installed, you should have access to the **getfacl** command, which displays the current ACLs of a file. For example, the following command displays the current ACLs for the anaconda-ks.cfg file in the /root directory:

```
# file: anaconda-ks.cfg
# owner: root
# group: root
user::rw-
group::---
other::---
```

Run the **ls -l /root/anaconda-ks.cfg** command. You should recognize every element of the ACLs shown here in the output. The ACLs that you'll add shortly are over and above those shown here. But first, you'll need to make a filesystem friendly to that second level of ACLs.

Make a Filesystem ACL Friendly

Before a file or directory can be configured with ACLs, you need to mount the associated filesystem with the same attribute. If you're just testing a system for ACL, you can remount an existing partition appropriately. For example, if /home is mounted on /dev/sda3, I can remount it with ACL using the following command:

```
# mount -o remount -o acl /dev/sda3 /home
```

To make sure this is the way /home is mounted on the next reboot, edit /etc/fstab. Based on the previous command, the associated line might read as follows:

```
/dev/sda3     /home     ext3      defaults,acl      1,2
```

In most cases on RHEL 6, you'll see **UUID=*somelargehexidecimalnumber*** in place of the device file. Once the change is made to /etc/fstab, you can activate it with the following command:

```
# mount -o remount /home
```

To confirm that the /home directory is mounted with the acl option, run the **mount** command alone, without switches or options. You should see acl in the output similar to what's shown here:

```
/dev/sda3 on /home type ext4 (rw,acl)
```

Now you can start working with ACL commands to set secondary access controls on desired files and directories.

Manage ACLs on a File

Now with a properly mounted filesystem and appropriate permissions, you can manage ACLs on a system. To review default ACLs, run the getfacl ***filename*** command. For this example, I've created a text file named TheAnswers in the /home/examprep directory. The following is the output from the getfacl /home /examprep/TheAnswers command:

```
# file home/examprep/TheAnswers
# owner: examprep
# group: proctors
user::rw-
group::r--
other::---
```

Note that the TheAnswers file is owned by user examprep and group proctors. That user owner has read and write permissions; that group owner has read permissions to that file. In other words, while the examprep user can change the Answers file, user members of the proctors group can read the TheAnswers file.

Now if you were the examprep or the root user on this system, you could assign ACLs for the file named TheAnswers for myself (user michael) with the **setfacl** command. For example, the following command gives me read permissions to that file:

```
# setfacl -m u:michael:rwx /home/examprep/TheAnswers
```

This command modifies the ACLs for the noted file, modifying (**-m**) the ACLs for user michael, giving that user read, write, and execute permissions to that file. To confirm, run the **getfacl** command on that file as shown in Figure 4-2.

But when I try to access that file from my user account, it doesn't work. Actually, if I try to access the file with the vi text editor, it suggests that the file /home/examprep/TheAnswers is a new file. And then it refuses to save any changes I might make to that file.

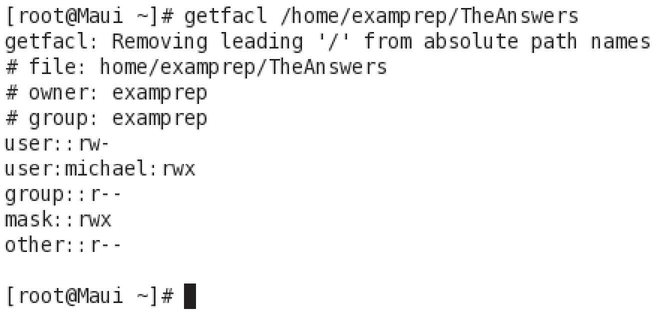
Before files from the /home/examprep directory are accessible, the administrative user will need to either change the permissions or the ACL settings associated with that directory. Before we get to modifying discretionary access controls on a directory, let's explore some different setfacl commands.

Despite the name, the **setfacl** command can be used to remove such ACL privileges with the **-x** switch. For example, the following command deletes the previously configured rwx privileges for user michael:

```
# setfacl -x u:michael /home/examprep/TheAnswers
```

FIGURE 4-2

The ACLs of a file



```
[root@Maui ~]# getfacl /home/examprep/TheAnswers
getfacl: Removing leading '/' from absolute path names
# file: home/examprep/TheAnswers
# owner: examprep
# group: examprep
user::rw-
user:michael:rwx
group::r--
mask::rwx
other::r--

[root@Maui ~]# 
```

In addition, the **setfacl** command can be used with groups; for example, if the teachers group exists, the following command would give read privileges to users who are members of that group:

```
# setfacl -m g:teachers:r-- /home/examprep/TheAnswers
```

If you want to see how ACLs work, don't remove the ACL privileges on the TheAnswers file, at least not yet. Alternatively, if you want to start over, the following command, with the **-b** switch, removes all ACL entries on the noted file.

```
# setfacl -b /home/examprep/TheAnswers
```

Some of the switches available for the **setfacl** command are shown in Table 4-3:

One slightly dangerous option relates to other users. For example, the following command:

```
# setfacl -m o:rwx /home/examprep/TheAnswers
```

allows other users read, write, and execute permissions for the TheAnswers file. It does so by changing the primary permissions for the file, as shown in the output to the **ls -l /home/examprep/TheAnswers** command. The **-b** and the **-x** switches don't remove such changes; you'd have to use one of the following commands:

```
# setfacl -m o:--- /home/examprep/TheAnswers

# chmod o-rwx /home/examprep/TheAnswers
```

TABLE 4-3

Description of File Permissions

Switch	Description
-b (--remove-all)	Removes all ACL entries; retains standard ugo/rwx permissions
-k	Deletes default ACL entries
-m	Modifies the ACL of a file, normally with a specific user (u) or group (g)
-n (--mask)	Omits the mask in recalculating permissions
-R	Applies changes recursively
-x	Removes a specific ACL entry

Configure a Directory for ACLs

There are two ways to set up a directory for ACLs. First, you could set the regular execute bit for all other users. One way to do so on the noted directory is with the following command:

```
# chmod 701 /home/examprep
```

It is a minimal way to provide access to files in a directory. Users other than examprep and root can't list the files in that directory. They have to know that the file TheAnswers actually exists to access that file.

However, with the execute bit set for other users, any user can access files in the /home/examprep directory for which he has permission. That should raise a security flag. Any user? Even though the file is hidden, do you ever want to give real privileges to anything to all users? Sure, ACLs have been set for only the TheAnswers file in that /home/examprep directory, but that's one layer of security that you've taken down voluntarily.

The right approach is to apply the **setfacl** command to the /home/examprep directory. The following command provides execute permissions for control associated with regular permissions is limited; the right ACLs can allow administrators to grant read, write, and execute access to a variety of users and groups. So the safest way to set up sharing is to set ACL execute permissions just for the user michael account on the noted directory, with the following command:

```
# setfacl -m u:michael:x /home/examprep
```

As the examprep user is the owner of the /home/examprep directory, that user can also run the noted **setfacl** command.

Sometimes, you may want to apply such ACLs to all files in a directory. In that case, the **-R** switch can be used to apply changes recursively; for example, the following command allows user michael to have read and execute permissions on all files in the /home/examprep directory as well as any subdirectories that may exist:

```
# setfacl -R -m u:michael:r-x /home/examprep
```

There are two methods available to unset these options. First, you could apply the **-x** switch to the previous command, omitting the permission settings:

```
# setfacl -R -x u:michael /home/examprep
```

Alternatively, you could use the **-b** switch; however, that would erase the ACLs configured for all users on the noted directory (and with the **-R** switch, applicable subdirectories):

```
# setfacl -R -b /home/examprep
```

Special Restrictions with ACLs

ACLs can also be used to limit permissions to specific users. For example, some RHEL 6 installations include a standard guest user for the GUI, xguest. For such systems, you may want to use ACLs to limit access to certain files or directories. For example, the following **setfacl** command can be used to limit access to the /etc/passwd file:

```
# setfacl -m u:xguest:--- /etc/passwd
```

While the /etc/passwd file does not normally include any passwords, it does include usernames. That's often a starting point for many crackers; malicious users with password cracking programs can then focus their efforts on standard users with known weaknesses. Without an xguest user, this command leads to a slightly confusing error message. If desired, you can test this by installing the xguest package, or by substituting a different username.

Of course, such actions can be applied recursively; if you don't want to allow user xguest access to any files in the /etc/ directory tree, the following command applies the aforementioned changes recursively:

```
# setfacl -R -m u:xguest:--- /etc
```

To test the result, try the **getfacl** command on a file somewhere down the /etc directory tree. For example, the **getfacl /etc/httpd/conf/httpd.conf** command examines the ACLs associated with the primary configuration file for the Apache web server. Note the difference between regular users and the xguest user in the following output:

```
# file: etc/httpd/conf/httpd.conf
# owner: root
# group: root
user::rw-
user:xguest:---
group::r--
mask::r--
other::r--
```

The user owner has read and write permissions. All other regular users have read permissions. But the xguest user has no permissions to do anything with the noted file.

As with individual files, the changes to the ACLs for the xguest user can be canceled with the **-x** switch. For example, the following command cancels ACL settings for that user recursively:

```
# setfacl -R -x u:xguest /etc
```

However, with ACLs, you can't deny access for a user to his home directory.

ACLs and Masks

The mask associated with an ACL limits the permissions available on a file. The mask shown in Figure 4-2 is rwx, which means there are no limits. If it were set to r, then the only permissions that could be granted with a command like **setfacl** is read. To change the mask on the TheAnswers file to read-only, run the following command:

```
# setfacl -m mask:r-- /home/examprep/TheAnswers
```

Now review the result with the **getfacl /home/examprep/TheAnswers** command. Pay attention to the entry for a specific user. Based on the ACL privileges given to user michael earlier, you'll see a difference with Figure 4-2:

```
user:michael:rwx    #effective:r--
```

In other words, with a mask of **r--**, you can try to provide other users with all the privileges in the world. But all that can be set with that mask is read privileges.

EXERCISE 4-3

Use ACLs to Deny a User

In this exercise, you'll set up ACLs to deny access to the loopback configuration file to a regular user. That is the ifcfg-lo file in the /etc/sysconfig/network-scripts directory. This exercise assumes that you've configured a regular user. As I've configured user michael on my systems, that is the regular user listed in this exercise. Substitute accordingly. To deny such access, take the following steps:

1. Back up a copy of the current configuration file for the loopback device. It's the ifcfg-lo file in the /etc/sysconfig/network-scripts directory. (Hint: use the **cp** and not the **mv** command.)

2. Execute the **setfacl -m u:michael:--- /etc/sysconfig/network-scripts/ifcfg-lo** command.
3. Review the results. Run the **getfacl** command on both copies of the file, in the /etc/sysconfig/network-scripts and the backup directories. What are the differences?
4. Log in as the target user. From the root administrative account, one method to do so is with the **su - michael** command.
5. Try to read the **/etc/sysconfig/network-scripts/ifcfg-lo** file in the vi text editor or even with the **cat** command. What happens?
6. Repeat the preceding step with the file in the backup directory. What happens?
7. Now run the **cp** command from the backup of the ifcfg-lo file, and overwrite the version in the /etc/sysconfig/network-scripts file. (Don't use the **mv** command for this purpose.) You may need to return as the root user to do so.
8. Try the **getfacl /etc/sysconfig/network-scripts/ifcfg-lo** command again. Are you surprised at the result?
9. There are two ways to restore the original ACL configuration for the ifcfg-lo file. First apply the **setfacl -b** command on the file. Did that work? Confirm with the **getfacl** command. If any other related commands have been applied, it may or may not have worked.
10. The only certain way to restore the original ACL of a file is to restore the backup, by first deleting the changed file in the /etc/sysconfig/network-scripts directory, and then by copying the file from the backup directory.
11. However, if you run Step 10, you may also need to restore the SELinux contexts of the file with the **restorecon -F /etc/sysconfig/network-scripts/ifcfg-lo** command. More information on the **restorecon** command is available later in this chapter.

NFS Shares and ACLs

While there's no evidence that the Red Hat exams will cover NFS-based ACLs, it is a feature that Linux administrators should know. As such, the description in this section just provides examples and is far from complete. One more complete description is available from IBM at www.ibm.com/developerworks/aix/library/au-filesys_NFSv4ACL/index.html.

Frequently, the /home directory is taken from a shared NFS directory. In fact, NFS-based ACLs are more fine-grained than standard ACLs. This feature was introduced with NFS version 4, the standard for RHEL 6. To that end, the **nfs4_getfacl** command can display the ACLs associated with files on a shared directory. Based on the ACLs previously given, Figure 4-3 shows the output to the **nfs4_getfacl** command.

The output is in the following format:

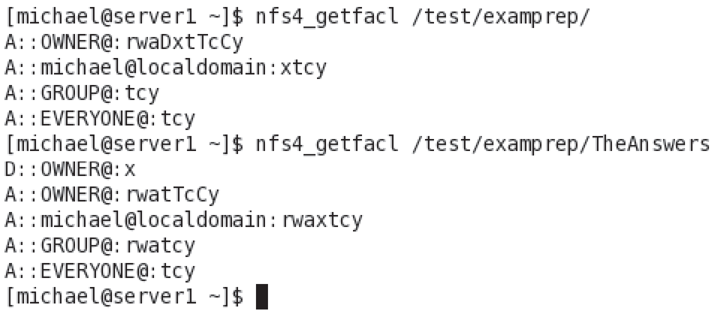
```
type:flags:principal:permissions
```

where the settings are delineated by the colon. Briefly, the two types shown either allow (**A**) or deny (**D**) the noted principal (a user or group) the specified permissions. No flags are shown in Figure 4-3, which can provide relatively fine-grained control. The principal may be a regular user or group, in lowercase. It may also be a generic user such as the file OWNER, the GROUP that owns the file, or other users, as specified by EVERYONE. The permissions as shown in Table 4-4, are more fine-grained. The effect varies depending on whether the object is a file or a directory.

The configuration of NFS as a client is covered in Chapter 6, with other local and network filesystems. The configuration of an NFS server is an RHCE objective covered in Chapter 16.

FIGURE 4-3

NFS version 4 ACLs



```
[michael@server1 ~]$ nfs4_getfacl /test/examprep/
A::OWNER@:rwaDxtTcCy
A::michael@localdomain:xtcy
A::GROUP@:tcy
A::EVERYONE@:tcy
[michael@server1 ~]$ nfs4_getfacl /test/examprep/TheAnswers
D::OWNER@:x
A::OWNER@:rwatTcCy
A::michael@localdomain:rwaxtcy
A::GROUP@:rwatcy
A::EVERYONE@:tcy
[michael@server1 ~]$ █
```

TABLE 4-4

Description of NFSv4 ACL Permissions

Permission	Description
r	Read file or list directory
w	Write to a file or create a new file in a directory
a	Append data to a file or create a subdirectory
x	Execute a script or change a directory
d	Delete the file or directory
D	Delete the subdirectory
t	Read the attributes of the file or directory
T	Write the attributes of the file or directory
c	Read the ACLs of the file or directory
C	Write the ACLs of the file or directory
y	Synchronize the file or directory

CERTIFICATION OBJECTIVE 4.03

Basic Firewall Control

Traditionally, firewalls were configured only between LANs and outside networks such as the Internet. But as security threats increase, there's an increasing need for firewalls on every system. RHEL 6 includes firewalls in every default configuration.

The best firewalls come in layers. They include packet filters with commands such as **iptables**. They include TCP Wrappers to control traffic to and from TCP-based services. They include controls from individual services. Arguably, they also include mandatory access control tools such as SELinux. While SELinux is covered in part later in this chapter, tools like TCP Wrappers and firewalls from individual services are covered in the RHCE portion of this book.

Before you send a message over a network, the message is broken down into smaller units called *packets*. Administrative information, including the type of data, the source address, and destination address, is added to each packet. The packets are reassembled when they reach the destination computer. A firewall examines these administrative fields in each packet to determine whether to allow the packet to pass.

There are RHCSA and RHCE requirements related to the **iptables** command. For the RHCSA, you need to understand how to configure a firewall to either block or allow network communication through one or more ports. For the RHCE, you need to know how to use the iptables command to filter packets based on elements such as source and destination IP addresses.

on the job

RHEL 6 also includes a firewall command for IPv6 networks,* ip6tables*. The associated commands are almost identical. Unlike* iptables*, the* ip6tables *command is not listed in the Red Hat objectives.

Standard Ports

Linux communicates over a network, primarily using the TCP/IP protocol suite. Different protocols use certain ports and protocols by default, as defined in the /etc/services file. It may be useful to know some of these ports by heart, such as those described in Table 4-5. Be aware, some of these ports may communicate using one or more of the following protocols: the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), and the Internet Control Message Protocol (ICMP). Such communications are listed in the /etc/services file. For example, as noted in the following excerpts from the /etc/services file, communications to FTP servers may proceed using both TCP and UDP protocols.

on the job

Strictly speaking, "transport level" protocols other than TCP, UDP, and ICMP may be specified with the* iptables *command. For example, the Encapsulating Security Payload (ESP) and the Authentication Header (AH) protocols are used with the Internet Protocol Security (IPsec) suite.

```
ftp   21/tcp
ftp   21/udp
```

However, you'll see shortly that the Red Hat firewall configuration tools open only TCP communications for FTP services, and the default vsFTP server configured in Chapter 1 works fine under such circumstances.

TABLE 4-5

Common TCP/IP Ports

Port	Description
21	FTP
22	Secure Shell (SSH)
23	Telnet
25	Simple Mail Transfer Protocol (SMTP), e.g., Postfix, sendmail
53	Domain Name Service servers
80	Hypertext Transfer Protocol (HTTP)
88	Kerberos
110	Post Office Protocol, version 3 (POP3)
139	Network Basic Input/Output System (NetBIOS) session service
143	Internet Mail Access Protocol (IMAP)
443	HTTP, secure (HTTPS)

A Focus on iptables

The philosophy behind **iptables** is based on "chains." These are sets of rules applied to each network packet, chained together. Each rule does two things: it specifies the conditions a packet must meet to match the rule, and it specifies the action if the packet matches.

The **iptables** command uses the following basic format:

```
iptables -t tabletype <action direction> <packet pattern> -j <what to do>
```

Now analyze this command, step by step. First is the ***-t tabletype*** switch. There are two basic *tabletype* options for **iptables**:

- **filter** Sets a rule for filtering packets.
- **nat** Configures Network Address Translation, also known as masquerading, discussed later in this chapter.

The default is **filter**; if you don't specify a ***-t tabletype,*** the **iptables** command assumes that the command is applied as a packet filter rule.

Next is the <***action direction***>. There are four basic actions associated with **iptables** rules:

- **-A (--append)** Appends a rule to the end of a chain.

- **-D (--delete)** Deletes a rule from a chain. Specify the rule by the number or the packet pattern.
- **-L (--list)** Lists the currently configured rules in the chain.
- **-F (--flush)** Flushes all of the rules in the current **iptables** chain.

☆f you're appending to (**-A**) or deleting from (**-D**) a chain, you'll want to apply it to network data traveling in one of three directions:

- **INPUT** All incoming packets are checked against the rules in this chain.
- **OUTPUT** All outgoing packets are checked against the rules in this chain.
- **FORWARD** All packets being sent to another computer are checked against the rules in this chain.

Typically, each of these directions is the name of a chain.

Next, you need to configure a <***packet pattern***>. All **iptables** firewalls check every packet against this pattern. The simplest pattern is by IP address:

- **-s *ip_address*** All packets are checked for a specific source IP address.
- **-d *ip_address*** All packets are checked for a specific destination IP address.

Packet patterns can be more complex. In TCP/IP, packets are transported using the TCP, UDP, or ICMP protocol. You can specify the protocol with the **-p** switch, followed by the destination port (**--dport**). For example, the **-p tcp --dport 80** extension affects users outside your network who are trying to use an HTTP connection.

Once the **iptables** command finds a packet pattern match, it needs to know what to do with that packet, which leads to the last part of the command, **-j <*what to do*>**. There are three basic options:

- **DROP** The packet is dropped. No message is sent to the requesting computer.
- **REJECT** The packet is dropped. An error message is sent to the requesting computer.
- **ACCEPT** The packet is allowed to proceed as specified with the **-A** action: **INPUT**, **OUTPUT**, or **FORWARD**.

Take a look at some examples of how you can use **iptables** commands to configure a firewall. The first step is always to see what is currently configured, with the following command:

```
# iptables -L
```

If **iptables** is properly configured, it should return chain rules in three different categories: **INPUT**, **FORWARD**, and **OUTPUT**. More examples are described in Chapter 10.

Keep That Firewall in Operation

Linux firewalls based on the **iptables** command are based on the service of the same name. To review current rules, run the **iptables -L** command. Suppose all you see is the following blank list of rules:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

The **iptables** service may not be running. Make sure to start it, and to make sure firewalls are running after the next reboot, run the following commands:

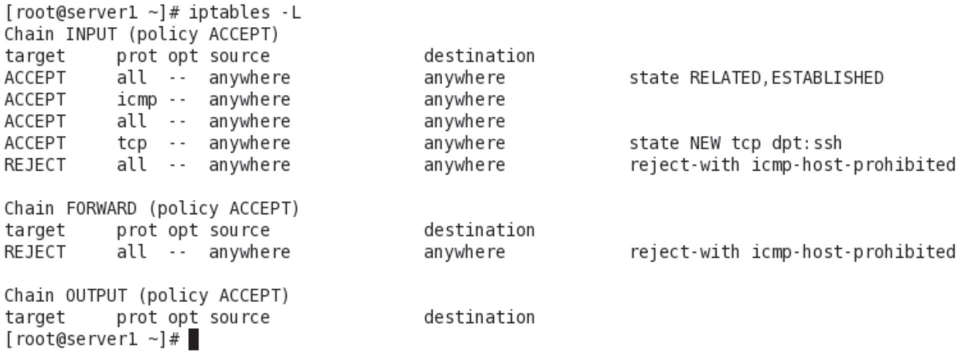
```
# /etc/init.d/iptables start
# chkconfig iptables on
```

The rules used by a Red Hat firewall are based on the /etc/sysconfig/iptables file, described in the next section.

The Default RHEL 6 Firewall

The current RHEL 6 firewall is shown in the output to the **iptables -L** command. The output on the default server1.example.com system is shown in Figure 4-4.

Firewall rules are divided into three categories, based on the direction of the data. INPUT rules are applied to data packets destined for the local system. FORWARD rules limit data going through the local system to another system. OUTPUT rules may limit data that goes out from the local system.



```
[root@server1 ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere            state NEW tcp dpt:ssh
REJECT     all  --  anywhere             anywhere            reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere            reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@server1 ~]# █
```

FIGURE 4-4

Default RHEL 6 firewall rules

Six columns of information are shown in Figure 4-4, which correspond to various **iptables** command options. The firewall shown is based on the following rules listed in the /etc/sysconfig/iptables file. The first line specifies that the rules to follow are filtering rules. Alternative rules support Network Address Translation (NAT) or mangling; NAT is discussed in Chapter 10.

```
*filter
```

Next, network traffic that is directed to the local system, intended to be forwarded, and are sent out, are normally accepted by default with the ACCEPT option.

```
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
```

Some security professionals, including the U.S. National Security Agency, suggest that the ACCEPT should be changed to DROP for at least the INPUT and FORWARD lines. If you accept this recommendation, packets that aren't explicitly accepted by other rules are automatically dropped. However, that level of security may be covered in Red Hat's Enterprise Security Network Services course (RHS333), which is open to those who have already passed the RHCE exam. The [0:0] are byte and packet counts, which each start at 0.

The lines that follow are all applied to the **iptables** command. Every switch and option listed in this file should be available on the associated man page.

The next line keeps current network communications going. The ESTABLISHED option continues to accept packets on current network connections. The RELATED option accepts packets for follow-on network connections, such as for FTP data transfers.

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

The next connection accepts packets associated with ICMP, most commonly associated with the **ping** command. When a packet is rejected, the associated message also uses the ICMP protocol.

```
-A INPUT -p icmp -j ACCEPT
```

That may raise a warning flag, as the "ping of death" is one common attack. You could block or limit responses to the **ping** command using some of the filtering discussed in Chapter 10.

The following line adds (**-A**) a rule to an INPUT chain, associated with the network interface (**-i**) known as the loopback adapter (lo). Any data processed through that device jumps (**-j**) to acceptance.

```
-A INPUT -i lo -j ACCEPT
```

The next line is the only one that directly accepts new regular network data, using the TCP protocol, over anything but the loopback adapter. It looks for a match (**-m**) for a NEW connection state (**state --state**), for matching TCP packets, using the TCP protocol (**-p tcp**), sent to a destination port (**--dport**) of 22. Network packets that meet all of these criteria are accepted (**-j ACCEPT**). Once the connection is established, the first regular rule described in this chapter continues to accept packets on that established connection.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

The last two rules reject all other packets, with an **icmp-host-prohibited** message sent to the originating system.

```
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```

The COMMIT ends the list of rules.

```
COMMIT
```

As this is the section associated with the RHCSA exam, a more detailed discussion can be found in Chapter 10, for the RHCE exam. At this level, you need to know how to manage these firewalls with the standard configuration tools provided.

The Firewall Configuration Tools

You can automate the process of configuring a firewall. For that purpose, RHEL includes both console and GUI versions of the Firewall Configuration tool. In this case, both tools are just filled with about the same number of features. While the look and feel of the two tools are different, the basic capabilities with respect to configuring access for trusted services are the same.

If you want to experiment with the Firewall Configuration tools, it's a good practice to first back up the associated configuration file, /etc/sysconfig/iptables. If mistakes are made, restore the original version of the file and run the following command:

```
# /etc/init.d/iptables restart
```

To learn about the changes that are made, compare the resulting /etc/sysconfig/iptables file with the backup. Any differences will be based on what you do in the Firewall Configuration tool.

exam watch

If you use a Firewall Configuration tool, you may want to avoid editing the /etc/sysconfig/iptables file directly. Any changes will just be overwritten the next time you use that tool. Chapter 10 describes how you can set up customized rules with the Firewall Configuration tool.

Trusted Services for Firewall Configuration Tools

Whichever tool is selected, both support easy configuration of a firewall to allow access to a variety of servers described in Table 4-6.

When comparing Table 4-6 to Table 4-5, you might note that while both TCP and UDP protocols are frequently reserved on a port for a specific service, frequently only one of these protocols is used by the actual service.

The Console Firewall Configuration Tool

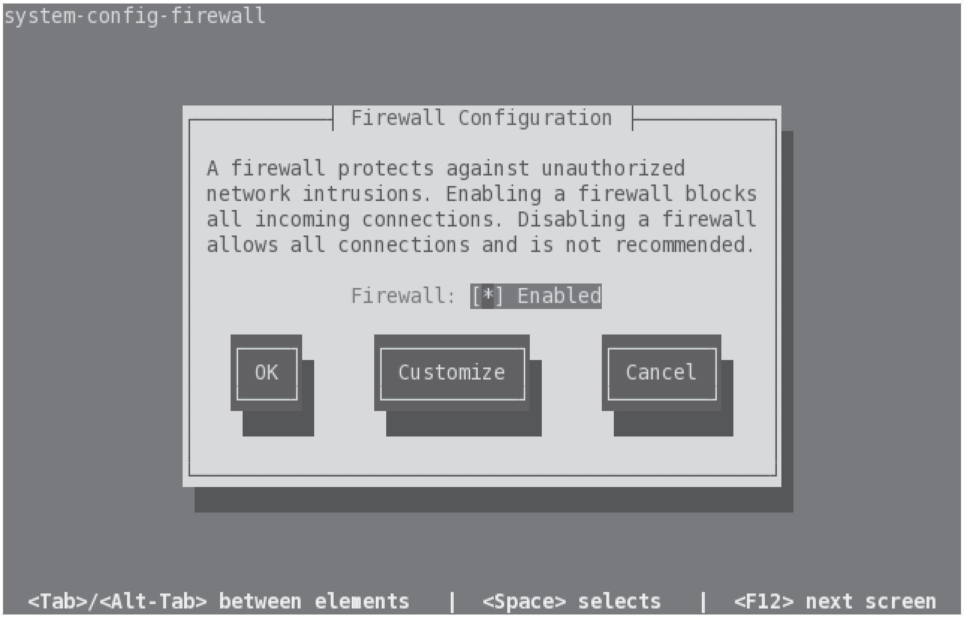
You can start the console firewall configuration tool with the **system-config-firewall-tui** command; the result is shown in Figure 4-5. As shown in the figure, firewalls are enabled by default; the option can be deselected.

TABLE 4-6 Common TCP/IP Ports

Service	Description
Amanda Backup Client	A client associated with the Advanced Maryland Automatic Network Disk Archiver (AMANDA), associated with UDP port 10080
Bacula	An open-source network backup server; associated with TCP ports 9101, 9102, and 9103
Bacula client	Client for the Bacula server; associated with TCP port 9102
DNS	Domain Name Service (DNS) server; associated with port 53, using both TCP and UDP protocols
FTP	File Transfer Protocol (FTP) server, associated with TCP port 21
IMAP over SSL	IMAP over the Secure Sockets Layer (SSL) normally uses TCP port 993
IPsec	Associated with UDP port 500 for the Internet Security Association and Key Management Protocol (ISAKMP), along with the ESP and AH transport-level protocols
Mail (SMTP)	Simple Mail Transport Protocol server, such as sendmail or Postfix, using TCP port 25
Multicast DNS (mDNS)	Associated with UDP port 5353 to support the Linux implementation of zero configuration networking (zeroconf), known as Avahi
NFS4	NFS version 4 uses TCP port 2049, among others
Network Printing Client	The standard print client uses UDP port 631, based on the Internet Print Protocol (IPP)
Network Printing Server	The standard print server client uses TCP and UDP ports 631, based on the Internet Print Protocol (IPP)
OpenVPN	The open-source Virtual Private Network system, which uses UDP port 1194
POP-3 over SSL	POP-3 over the Secure Sockets Layer (SSL) normally uses TCP port 995
RADIUS	The Remote Authentication Dial In User Service (RADIUS) protocol uses UDP ports 1812 and 1813
Red Hat Cluster Suite	The Red Hat suite for multiple systems uses TCP ports 11111 and 21064, along with UDP ports 5404 and 5405

TABLE 4-6 Common TCP/IP Ports (*continued*)

Service	Description
Samba	The Linux protocol for communication on Microsoft networks uses TCP ports 139 and 445, along with UDP ports 137 and 138
Samba Client	The Linux protocol for client communication on Microsoft networks uses UDP ports 137 and 138
Secure WWW (HTTPS)	Communications to a secure web server uses TCP port 443
SSH	The SSH server uses TCP port 22
TFTP	Communications with the Trivial File Transfer Protocol (TFTP) server requires TCP port 69
TFTP Client	Strangely enough, no open port is required for a TFTP client; all communications proceed over the open TCP port 69 through the TFTP server
Virtual Machine Management	Remote access to KVM-based VMs use TCP port 16509
Virtual Machine Management (TLS)	Remote access to KVM-based VMs use TCP port 16509 can be configured with Transport Layer Security (TLS)
WWW (HTTP)	The well-known web server uses TCP port 80



```
system-config-firewall

                 ┤ Firewall Configuration ├

     A firewall protects against unauthorized
     network intrusions. Enabling a firewall blocks
     all incoming connections. Disabling a firewall
     allows all connections and is not recommended.

               Firewall: [*] Enabled

      OK          Customize          Cancel

 <Tab>/<Alt-Tab> between elements   |  <Space> selects   |  <F12> next screen
```

FIGURE 4-5 The Console Firewall Configuration tool

Press the TAB key as needed to highlight the Customize option and press ENTER to continue. Review the list of trusted services; while the order varies slightly, they match the list shown in Table 4-6. If you've configured one of these services on the local system, you'll want to activate the option to set them as trusted services. Scroll down until you see SSH. It should be enabled by default. Make appropriate selections and select Forward to continue.

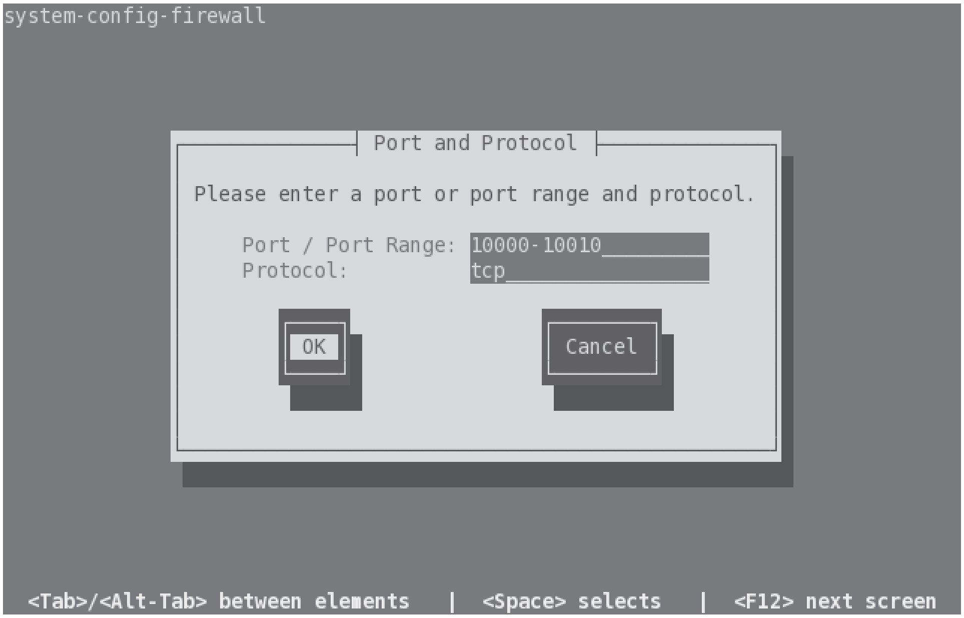
In the Other Ports window, highlight Add and press ENTER to open the Port and Protocol screen shown in Figure 4-6. The entries in that figure demonstrate how you might open up ports 10000 through 10010 under the TCP protocol in a firewall. Make any desired changes and select OK or abort by selecting Cancel.

Back in the Other Ports window, you can configure additional ports. If desired, you can click Forward to enter the Trusted Interfaces window, and then configure Masquerading, Port Forwarding, and ICMP filtering, but those details are more closely associated with the RHCE exam objectives discussed in Chapter 10.

Instead, select Close and press ENTER to return to the screen shown in Figure 4-5. You can repeat the Customize process again, select Cancel to disable the changes, or select OK to implement them. Before implementing changes, you're given a warning as shown in Figure 4-7.

FIGURE 4-6

Configure other ports for the firewall

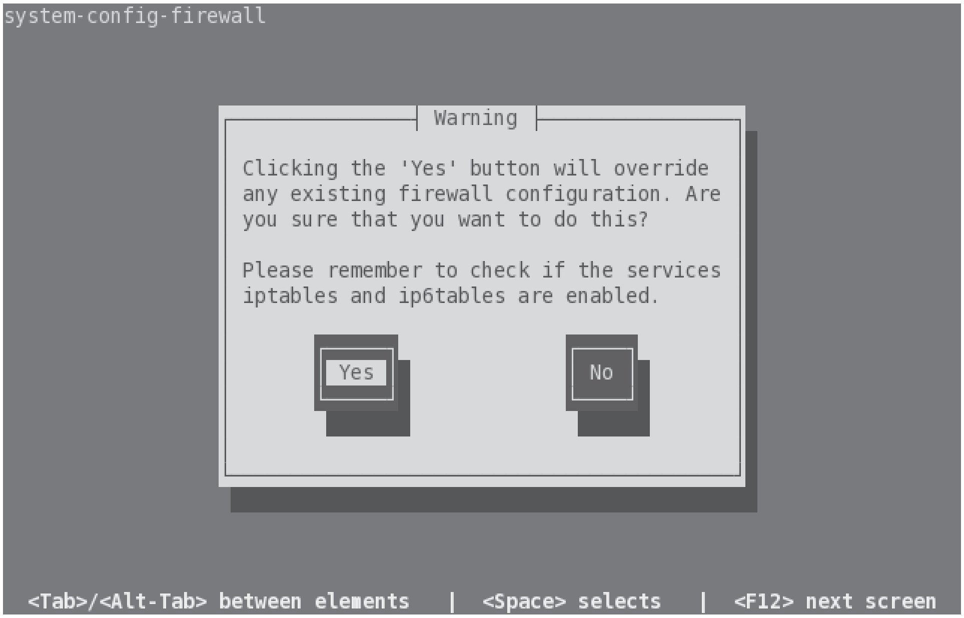


Please enter a port or port range and protocol.

Port / Port Range: 10000-10010

Protocol: tcp

<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen



```
system-config-firewall

Warning

Clicking the 'Yes' button will override
any existing firewall configuration. Are
you sure that you want to do this?

Please remember to check if the services
iptables and ip6tables are enabled.
```

Yes
No
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

FIGURE 4-7

Before implementing a new firewall

The GUI Firewall Configuration Tool

If the system-config-firewall package is installed, you can start the GUI Firewall Configuration tool in a GUI-based command line with the **system-config-firewall** command. Alternatively, in the GNOME Desktop Environment, click System | Administration | Firewall. The result is shown in Figure 4-8. As shown in the figure, firewalls are enabled by default, but they can be disabled by pressing the Disable button.

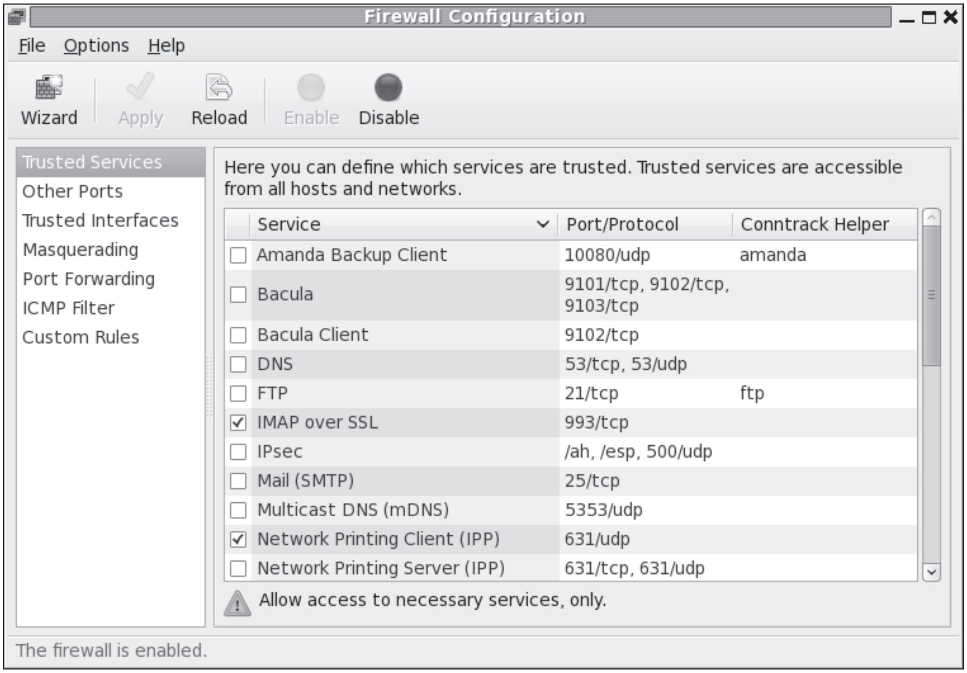
The GUI Firewall Configuration tool has the same list of services as the corresponding console-based tool, in a slightly different order.

If desired, you can also configure custom access ports; click Other Ports and click Add to open the Port and Protocol window shown in Figure 4-9. Note how the window prompts for ports based on the contents of the /etc/services file.

As with the console-based tool, the other features of the GUI Firewall Configuration tool are more closely associated with the RHCE exam. For more information, see Chapter 10.

FIGURE 4-8

The GUI Firewall Configuration tool



Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

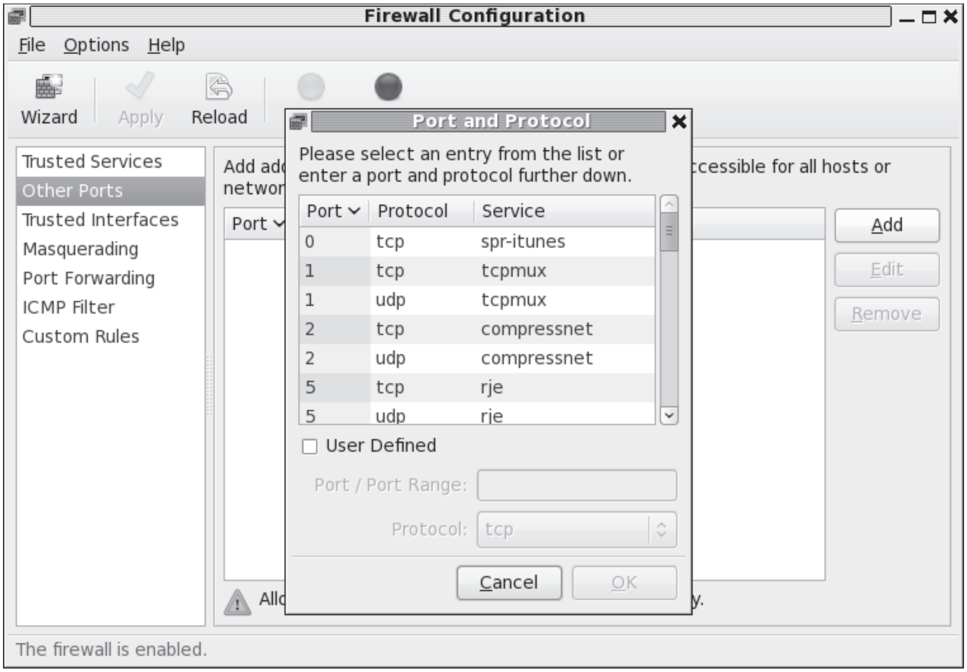
	Service	Port/Protocol	Conntrack Helper
□	Amanda Backup Client	10080/udp	amanda
□	Bacula	9101/tcp, 9102/tcp, 9103/tcp	
□	Bacula Client	9102/tcp	
□	DNS	53/tcp, 53/udp	
□	FTP	21/tcp	ftp
☑	IMAP over SSL	993/tcp	
□	IPsec	/ah, /esp, 500/udp	
□	Mail (SMTP)	25/tcp	
□	Multicast DNS (mDNS)	5353/udp	
☑	Network Printing Client (IPP)	631/udp	
□	Network Printing Server (IPP)	631/tcp, 631/udp	

Allow access to necessary services, only.

The firewall is enabled.

FIGURE 4-9

Adding custom ports in the GUI Firewall Configuration tool



Please select an entry from the list or enter a port and protocol further down.

Port	Protocol	Service
0	tcp	spr-itunes
1	tcp	tcpmux
1	udp	tcpmux
2	tcp	compressnet
2	udp	compressnet
5	tcp	rje
5	udp	rje

□ User Defined

Port / Port Range:

Protocol: tcp

The firewall is enabled.

EXERCISE 4-3

Adjust Firewall Settings

In this exercise, you'll adjust firewalls from the command line interface, and review the results with the nmap and telnet commands. While it does not matter how you address a problem on a Red Hat exam, in this exercise, you'll see what happens when the /etc/sysconfig/iptables configuration file is modified. Of course, it's possible to use the Firewall Configuration tool to perform the same tasks. This assumes a system with the default firewall described in this chapter.

1. Review the current active services on the local system with the **nmap localhost** command. Note the IP address of the local system with the **ifconfig eth0** command. If the local system is server1.example.com, that IP address should be 192.168.122.50.
2. Back up a copy of the current firewall configuration file, /etc/sysconfig /iptables. Make sure to use the **cp** and not the **mv** command.
3. Make sure the firewall is currently operational with the **/etc/sysconfig /iptables restart** command.
4. Go to a different system. You can do so from a different virtual machine, or you can access it remotely with the **ssh** command. If the tester1.example.com system is running, you can log into that system with the **ssh 192.168.122.150** command.
5. Use the **nmap** command to review what is seen through the firewall; for the noted server1.example.com system, the right command would be **nmap 192.168.122.50**; if the IP address found from Step 1 is different, substitute accordingly.
6. Return to the original system. Open the /etc/sysconfig/iptables file in a text editor.
7. Substitute 25 for 22 in the file, and save the changes.
8. Restart the firewall as was done in Step 3.
9. Navigate back to the tester1.example.com system as was done in Step 4.
10. Repeat Step 5. What do you see?
11. Return to the original system. If desired, restore the iptables file from backup, and restart the firewall as was done in Step 3.

CERTIFICATION OBJECTIVE 4.04

A Security-Enhanced Linux Primer

Security-Enhanced Linux (SELinux) was developed by the U.S. National Security Agency to provide a level of mandatory access control. It goes beyond the discretionary access control associated with file permission and ACLs. In essence, SELinux limits the damage if there is a security breach. For example, if the system account associated with an FTP service is compromised, SELinux makes it more difficult to use that account to compromise other services.

Basic Features of SELinux

SELinux assigns different contexts to each file, known as *subjects*, *objects*, and *actions*. In the SELinux world, a subject is a process, such as a command in action, or an application such as the Apache web server in operation. An object is a file. An action is what may be done by the subject to the object.

For example, the Apache web server process can take objects such as web pages and display them for the clients of the world to see. That action is normally allowed in the RHEL 6 implementation of SELinux, as long as the object files have appropriate SELinux contexts.

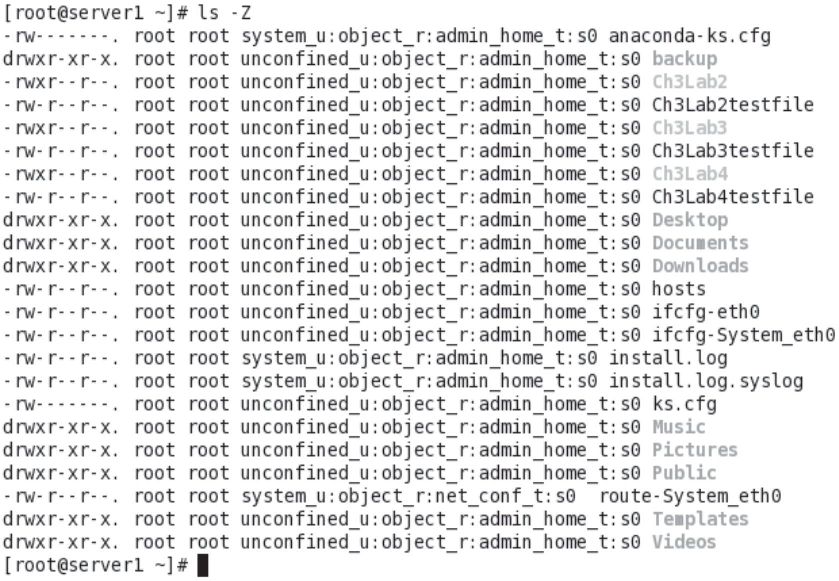
The contexts associated with SELinux are fine-grained. In other words, if a cracker breaks in and takes over your web server, SELinux contexts prevent that cracker from using that breach to break into other services.

To see the context of a particular file, run the **ls -Z** command. As an example, review what this command does in Figure 4-10, as it displays security contexts in my /root directory.

As noted at the beginning of this chapter, four objectives relate to SELinux on the RHCSA exam. You'll explore how to meet these objectives in the following sections.

SELinux Status

As suggested in the RHCSA objectives, you need to know how to "Set enforcing /permissive modes for SELinux." There are three available modes for SELinux: **enforcing**, **permissive**, and **disabled**. The **enforcing** and **disabled** modes are self-



```
[root@server1 ~]# ls -Z
-rw-------. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 backup
-rwxr--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab2
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab2testfile
-rwxr--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab3
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab3testfile
-rwxr--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab4
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 Ch3Lab4testfile
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Desktop
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Documents
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Downloads
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 hosts
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 ifcfg-eth0
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 ifcfg-System_eth0
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log.syslog
-rw-------. root root unconfined_u:object_r:admin_home_t:s0 ks.cfg
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Music
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Pictures
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Public
-rw-r--r--. root root system_u:object_r:net_conf_t:s0  route-System_eth0
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Templates
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Videos
[root@server1 ~]#
```

FIGURE 4-10 SELinux security contexts of different files

explanatory. SELinux in **permissive** mode means that any SELinux rules that are violated are logged, but the violation does not stop any action.

If you want to change the basic status of SELinux, change the **SELINUX** directive. The next time you reboot, the changes are applied to the system.

If SELinux is configured in **enforcing** mode, it protects systems in one of two ways: in **targeted** mode or in **mls** mode. The default is **targeted**, which allows you to customize what is protected by SELinux in a fine-grained manner. In contrast, MLS goes a step further, using the Bell-La Padula model developed for the Department of Defense. That model, as suggested in the /etc/selinux/targeted/setrans.conf file, supports layers of security between levels c0 and c3. While the c3 level is listed as “Top Secret,” the range of available levels goes all the way up to c1023. Such fine-grained levels of secrecy have yet to be fully developed. If you want to explore MLS, install the selinux-policy-mls RPM.

TABLE 4-7 Standard Directives in /etc/sysconfig/selinux

Directive	Description
SELINUX	Basic SELinux status; may be set to **enforcing**, **permissive**, or **disabled**.
SELINUXTYPE	Specifies the level of protection; set to **targeted** by default, where protection is limited to daemons. The alternative is **mls**, which is associated with Multi-Level Security (MLS).

exam watch

If you have to configure SELinux during an exam, it's no longer possible to do so during the installation process. SELinux is by default implemented in enforcing and targeted modes. If you have to configure SELinux and have to reboot, the process of applying SELinux policies can take several minutes. You won't be able to log in or do anything else during your exam. So plan ahead!

on the job

If you just want to experiment with SELinux, configure it in permissive mode. It'll log any violations without stopping anything. It's easy to set up with the SELinux Management tool, or you can set* SELINUX=permissive *in /etc/sysconfig/selinux. If the auditd service is running, violations are logged in the audit.log file in the /var/log/audit directory. Just remember, it's likely that Red Hat wants candidates to configure SELinux in enforcing mode during their exams.

SELinux Configuration at the Command Line

While SELinux is still under active development, it has become much more useful with the release of RHEL 6. Nevertheless, given the fear associated with SELinux, it may be more efficient to use the SELinux Administration tool to configure SELinux settings. And it's much improved from the GUI SELinux functionality that was part of the Security Level Configuration tool. You can even set SELinux contexts for individual directories.

To that end, the following sections show how you can configure and manage SELinux from the command line interface. However, as it's easier to demonstrate the full capabilities of SELinux using GUI tools, a detailed discussion of such capabilities will follow later in this chapter.

Configure Basic SELinux Settings

There are some essential commands that can be used to review and configure basic SELinux settings. To see the current status of SELinux, run the **getenforce** command; it returns one of three self-explanatory options: **enforcing**, **permissive**, or **disabled**. The **sestatus** command provides more information, with output similar to the following:

```
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

You can change the current SELinux status with the **setenforce** command; the options are straightforward:

```
# setenforce enforcing
# setenforce permissive
```

This changes the /selinux/enforce boolean. As booleans, you could substitute 1 and 0, respectively, for **enforcing** and **permissive**. However, changes to detailed SELinux booleans require different commands.

exam Watch

If SELinux is disabled, it may take a few minutes to reboot a system after setting SELinux in enforcing mode. However, the process is much less time-consuming than it was for RHEL 5.

Alternatively, if SELinux is disabled for some reason, the output would be:

```
SELinux status:     disabled
```

In that case, the **setenforce** command will not work. Instead, you'll have to set **SELINUX=enforcing** in the /etc/sysconfig /selinux file. And that requires a "relabel," where SELinux labels are applied to each file on the local system. That can take valuable time.

Configure Regular Users for SELinux

One change from RHEL 5 is the SELinux status of regular default users. To review the status of current users, run the **semanage login -l** command. Based on the default installation of RHEL 6, it leads to the following output:

```
__default__       unconfined_u          s0-s0:c0.c1023
root              unconfined_u          s0-s0:c0.c1023
system_u          system_u              s0-s0:c0.c1023
```

In other words, regular "default" users have the same SELinux privileges of the root administrative user. To confirm, run the **id -Z** command as a regular user. Without changes, it leads to the following output, which suggests that user is not confined by any SELinux settings.

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

While it may not be an exam requirement, regular users should be confined by SELinux. When user accounts are compromised, and they will be compromised, you want any damage that might be caused limited by SELinux rules. The following example confines adds (**-a**) regular user michael, specifying (**-s**) the user_u role for confinement:

```
# semanage login -a -s user_u michael
```

The user_u role should not have the ability to run the **su** or **sudo** commands described in Chapter 8. If desired, you can reverse the process with the **semanage -d michael** command. As user roles are still a work in progress, you should focus on the available user roles listed in the latest Red Hat documentation, as shown in Table 4-8.

One other commonly seen "user" role is system_u, which typically does not apply to regular users. It is a common user seen in the output to the **ls -Z** command for system and configuration files.

When a user role is changed, they don't take effect until the next login. For example, if I were to change the role for user michael to user_u in a GUI-based command line, the change would not take effect until I logged out and logged back in to the GUI. When I tried it on my system, I was no longer able to start any administrative configuration tools, and I did not have access to the **sudo** and **su** commands.

On some networks, you may want to change the role of future users to user_u. If you don't want regular users tinkering with administrative tools, you could make that change for future default users with the following command:

```
# semanage login -m -S targeted -s "user_u" -r s0 __default__
```

This command modifies (**-m**) the targeted policy store (**-S**), with SELinux user (**-s**) user_u, with the MLS s0 range (**-r**) for the default user. The "__default__" includes two underscore characters on each side of the word. As long as user_u is in

TABLE 4-8

Options for SELinux User Roles

User Role	Features
guest_u	No GUI, no networking, no access to the **su** or **sudo** commands
xguest_u	GUI, networking only via the Firefox web browser
user_u	GUI and networking available
staff_u	GUI, networking, and the **sudo** command available
unconfined_u	Full system access

effect for the default SELinux user, regular users won't have access to use administrative tools or commands like **su** and **sudo**. The following command reverses the process:

```
# semanage login -m -S targeted -s "unconfined_u" \
-r s0-s0:c0.c1023 __default__
```

The full MLS range is required (s0-s0:c0.c1023), as the unconfined_u user is not normally limited by MLS restrictions.

exam watch

MLS mode adds complexity to SELinux. Targeted mode with appropriate booleans and file contexts normally provides more than adequate security.

Manage SELinux Boolean Settings

Most SELinux settings are boolean—in other words, they're activated and deactivated by setting them to 1 or 0. Once set, the booleans are stored in the /selinux/booleans directory. One simple example is user_ping, which is normally set to 1, which allows users to run the **ping** command. Many of these SELinux settings are associated with specific RHCE services and will be covered in the second half of this book.

These settings can be read with the **getsebool** and modified with the **setsebool** commands. For example, the following output from the **getsebool allow_user_exec_content** command confirms that SELinux allows users to execute scripts either in their home directories or from the /tmp directory:

```
allow_user_exec_content --> on
```

This default applies to SELinux user_u users. In other words, with this boolean, such users can create and execute scripts in the noted directories. That boolean can be disabled either temporarily, or in a way that survives a reboot. One method for doing so is with the **setsebool** command. For example, the following command disables the noted boolean until the system is rebooted:

```
# setsebool allow_user_exec_content off
```

You can choose to substitute **=0** for **off** in the command. As this is a boolean setting, the effect is the same; the flag is switched off. However, the **-P** is required to make the change to the boolean setting survive a system reboot. Be aware, the changes don't take effect until the next time the affected user actually logs into the associated system.

A full list of available booleans is available in the output to the **getsebool -a** command.

For more information on each boolean, run the **semanage boolean -l** command. While the output includes descriptions of all available booleans, it is a database that can be searched with the help of the **grep** command.

List and Identify SELinux File Contexts

If you've enabled SELinux, the **ls -Z** command lists current SELinux file contexts, as shown earlier in Figure 4-10. As an example, take the relevant output for the anaconda-ks.cfg file from the /root directory:

```
-rw-------. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
```

The output includes the regular ugo/rwx ownership and permission data. It also specifies four elements of SELinux security: the user, role, type, and MLS level for the noted file. Generally, the SELinux user associated with a file is system_u or unconfined_u, and this generally does not affect access. In most cases, files are associated with an object_r, an object role for the file. It's certainly possible that future versions of SELinux will include more fine-grained options for the user and role.

The key file context is the type; in this case, admin_home_t. When you configured FTP and HTTP servers in Chapter 1, you changed the type of the configured directory and the files therein to match the default type of shared files from those services with the **chcon** command.

For example, to configure a nonstandard directory for an FTP server, make sure the context matches the default FTP directory. Consider the following command:

```
# ls -Z /var/ftp/
drwxr-xr-x  root root system_u:object_r:public_content_t pub
```

The contexts are the system user (**system_u**) and system objects (**object_r**), for type sharing with the public (**public_content_t**). If you create another directory for FTP service, you'll need to assign the same security contexts to that directory. For example, if you create an /ftp directory as the root user and run the **ls -Zd /ftp** command, you'll see the contexts associated with the /ftp directory as shown:

```
drwxr-xr-x. root  root  unconfined_u:object_r:root_t   ftp
```

To change the context, use the **chcon** command. If there are subdirectories, you'll want to make sure changes are made recursively with the **-R** switch. In this case, to change the user and type contexts to match /var/ftp, run the following command:

```
# chcon -R -u system_u -t public_content_t /ftp
```

If you want to support uploads to your FTP server, you'll have to assign a different type context, specifically **public_content_rw_t**. That corresponds to the following command:

```
# chcon -R -u system_u -t public_content_rw_t /ftp
```

But wait, in Chapter 1, you used a different variation on the **chcon** command. To use that lesson, the following command uses user, role, and context from the /var/ftp directory, and applies the changes recursively:

```
# chcon -R --reference-/var/ftp /ftp
```

Restore SELinux File Contexts

Default contexts are configured in /etc/selinux/targeted/contexts/files/file_contexts. If you make a mistake and want to restore the original SELinux settings for a file, the **restorecon** command restores those settings based on the file_contexts configuration file. However, the defaults in a directory may vary. For example, the following command (with the **-F** switch forcing changes) leads to a different set of contexts for the /ftp directory:

```
# restorecon -F /ftp
# ls -Zd /ftp
drwxr-xr-x.  root  root  system_u:object_r:default_t  ftp
```

You may notice that the user context is different from when the /ftp directory was created. That's due to the first line in the aforementioned file_contexts file, which applies the noted contexts:

```
/.*     system_u:object_r:default_t:s0
```

The file_contexts file is important for another reason. Any files and subdirectories created in listed directories inherit the file contexts associated with each directory. In other words, if you create a /srv/ftp directory, it inherits the var_t file context associated with that directory.

Identify SELinux Process Contexts

As discussed in Chapter 9, the **ps** command lists currently running processes. In a SELinux system, there are contexts for each running process. To see those contexts for all processes currently in operation, run the **ps -eZ** command, which lists every

(**-e**) process SELinux context (**-Z**). Figure 4-11 includes a varied excerpt from the output of that command on my system.

While the user and role don't change often, the process type varies widely, frequently matching the purpose of the running process. For example, from the top of the figure, you can see how the Hardware Authentication Layer Daemon (hald) is matched by the hald_t SELinux type. You should be able to identify how at least some of the other SELinux types matches the associated service.

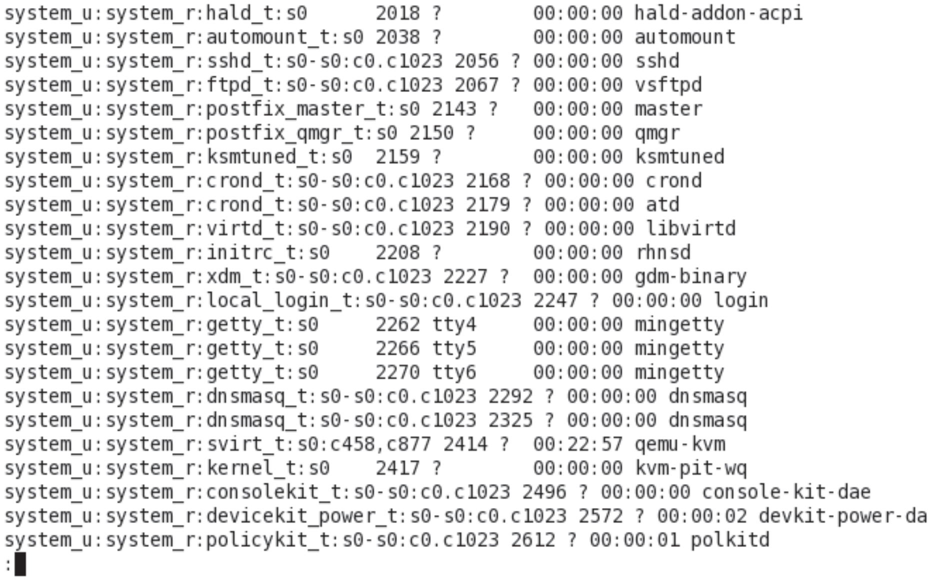
In other words, while there is a large variety of SELinux types, they're consistent with the running process.

Diagnose and Address SELinux Policy Violations

If there's a problem, SELinux is running in enforcing mode, and you're sure there are no problems with the target service or application, don't disable SELinux! Red Hat has made it easier to manage and troubleshoot. According to Red Hat, the top three causes of SELinux-related problems are: labeling, context, and boolean settings. As the first two relate to those contexts shown in the output to the **ls -Z** command, they are closely related.

FIGURE 4-11

SELinux security contexts of different processes



```
system_u:system_r:hald_t:s0      2018 ?        00:00:00 hald-addon-acpi
system_u:system_r:automount_t:s0 2038 ?        00:00:00 automount
system_u:system_r:sshd_t:s0-s0:c0.c1023 2056 ? 00:00:00 sshd
system_u:system_r:ftpd_t:s0-s0:c0.c1023 2067 ? 00:00:00 vsftpd
system_u:system_r:postfix_master_t:s0 2143 ?   00:00:00 master
system_u:system_r:postfix_qmgr_t:s0 2150 ?     00:00:00 qmgr
system_u:system_r:ksmtuned_t:s0  2159 ?        00:00:00 ksmtuned
system_u:system_r:crond_t:s0-s0:c0.c1023 2168 ? 00:00:00 crond
system_u:system_r:crond_t:s0-s0:c0.c1023 2179 ? 00:00:00 atd
system_u:system_r:virtd_t:s0-s0:c0.c1023 2190 ? 00:00:00 libvirtd
system_u:system_r:initrc_t:s0    2208 ?        00:00:00 rhnsd
system_u:system_r:xdm_t:s0-s0:c0.c1023 2227 ?  00:00:00 gdm-binary
system_u:system_r:local_login_t:s0-s0:c0.c1023 2247 ? 00:00:00 login
system_u:system_r:getty_t:s0     2262 tty4     00:00:00 mingetty
system_u:system_r:getty_t:s0     2266 tty5     00:00:00 mingetty
system_u:system_r:getty_t:s0     2270 tty6     00:00:00 mingetty
system_u:system_r:dnsmasq_t:s0-s0:c0.c1023 2292 ? 00:00:00 dnsmasq
system_u:system_r:dnsmasq_t:s0-s0:c0.c1023 2325 ? 00:00:00 dnsmasq
system_u:system_r:svirt_t:s0:c458,c877 2414 ?  00:22:57 qemu-kvm
system_u:system_r:kernel_t:s0    2417 ?        00:00:00 kvm-pit-wq
system_u:system_r:consolekit_t:s0-s0:c0.c1023 2496 ? 00:00:00 console-kit-dae
system_u:system_r:devicekit_power_t:s0-s0:c0.c1023 2572 ? 00:00:02 devkit-power-da
system_u:system_r:policykit_t:s0-s0:c0.c1023 2612 ? 00:00:01 polkitd
:
```

SELinux Audits

Problems with SELinux should be documented in the associated log file, audit.log in the /var/log/audit directory. The file may be confusing, especially the first time you read it. A number of tools are available to help decipher this log.

First, the audit search (**ausearch**) command can help filter for specific types of problems. For example, the following command lists all SELinux events associated with the use of the **sudo** command:

```
# ausearch -m avc -c sudo
```

Such events are known as Access Vector Cache (**-m avc**) messages; the **-c** allows you to specify the name commonly used in the log, such as httpd or su. If you've experimented with the user_u SELinux user described earlier in this chapter, there should be several related messages available from the audit.log file.

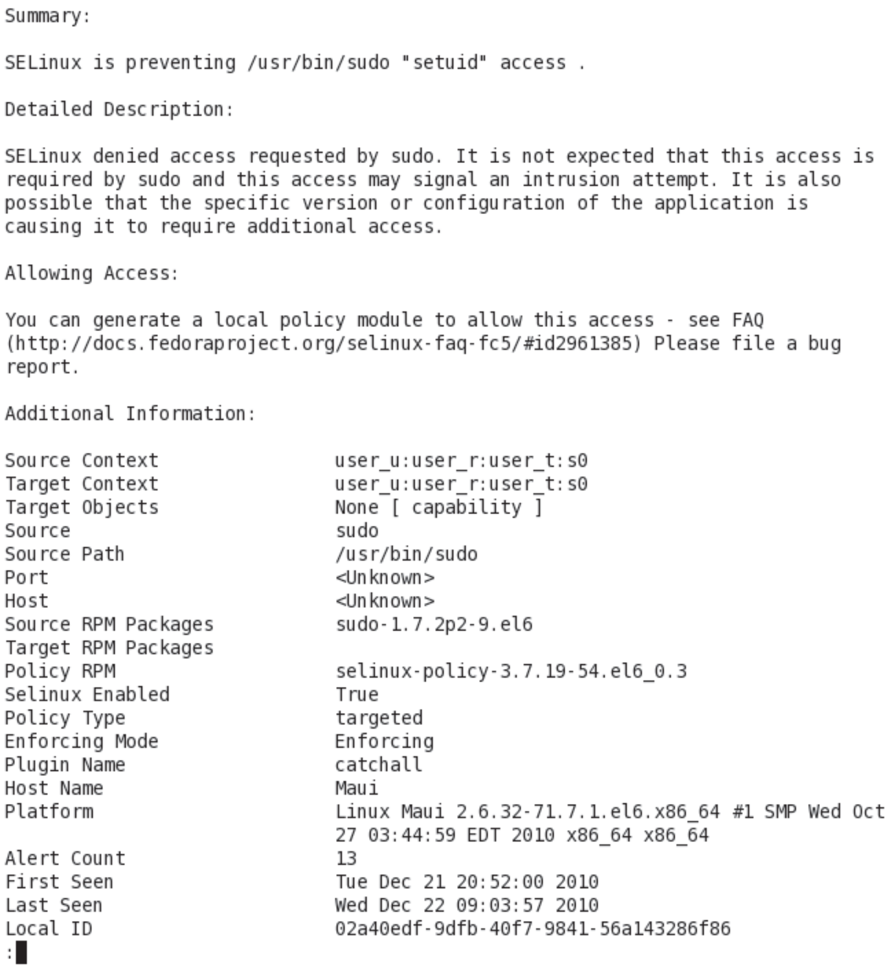
Even for most administrators, the output is still a lot of gobblygook. However, it should include identifying information such as the audited user ID (shown as auid), which can help you identify the offending user. Perhaps the user needs such access, perhaps that user's account has been compromised. In any case, the alert may cause you to pay more attention to that account.

In contrast, the **sealeart -a /var/log/audit/audit.log** command may provide more clarity. An excerpt is shown in Figure 4-12.

SELinux Label and Context Issues

Considering Figure 4-12 and the SELinux concepts described so far, you might wonder if the user in question is allowed to run the **sudo** command. If the problem were in the /etc/sudoers file covered in Chapter 8, the SELinux alert message might not even appear. So you should pay attention to the source and target contexts. As they match, the file context is not the issue.

By process of elimination, that points to the user context described earlier as the problem. The UID of the user in question should be listed later in the file, under "Raw Audit Messages". If the user in question requires access to the **sudo** command, you should change the context of that user with the **semanage login** command described earlier. Otherwise, the user might just be experimenting with Linux. Any access to the **sudo** command will be documented in the /var/log/secure log file. So if that user actually gets by the SELinux-based limits on the use of the **sudo** command, it'll be documented in that file.



```
Summary:

SELinux is preventing /usr/bin/sudo "setuid" access .

Detailed Description:

SELinux denied access requested by sudo. It is not expected that this access is
required by sudo and this access may signal an intrusion attempt. It is also
possible that the specific version or configuration of the application is
causing it to require additional access.

Allowing Access:

You can generate a local policy module to allow this access - see FAQ
(http://docs.fedoraproject.org/selinux-faq-fc5/#id2961385) Please file a bug
report.

Additional Information:

Source Context                user_u:user_r:user_t:s0
Target Context                user_u:user_r:user_t:s0
Target Objects                None [ capability ]
Source                        sudo
Source Path                   /usr/bin/sudo
Port                          <Unknown>
Host                          <Unknown>
Source RPM Packages           sudo-1.7.2p2-9.el6
Target RPM Packages
Policy RPM                    selinux-policy-3.7.19-54.el6_0.3
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Plugin Name                   catchall
Host Name                     Maui
Platform                      Linux Maui 2.6.32-71.7.1.el6.x86_64 #1 SMP Wed Oct
                              27 03:44:59 EDT 2010 x86_64 x86_64
Alert Count                   13
First Seen                    Tue Dec 21 20:52:00 2010
Last Seen                     Wed Dec 22 09:03:57 2010
Local ID                      02a40edf-9dfb-40f7-9841-56a143286f86
:█
```

FIGURE 4-12 One SELinux alert

SELinux Boolean Issues

After deactivating the allow_user_exec_content boolean described earlier, I created a simple script named script1 for a user governed by the user_u label. After making that script executable, I tried running it with the **/home/examprep/script1** command. Even though that user had ownership of the file, with executable permissions set, that attempt led to the following message:

```
-bash: /home/examprep/script1: Permission denied
```

That led to the log excerpt shown in Figure 4-13. Note the Fix Command section; it explicitly cites the command required to address the problem. As an administrator, you need to decide whether such users should be given the ability to execute their own scripts. If so, then the noted command would address the problem.

The GUI SELinux Management Tool

If you've taken the time to learn SELinux from the command line, this section should be just a review. For many users, the easiest way to change SELinux settings is with the SELinux Administration tool, which you can start with the **system-config-selinux** command. As shown in Figure 4-14, it starts with a basic view of the status of SELinux on the local system, reflecting some of the information shown in the output to the **sestatus** command.

exam watch

To install the GUI SELinux Management tool, run the* yum install policycoreutils-gui *command. In this case, there is no system-config-selinux package.

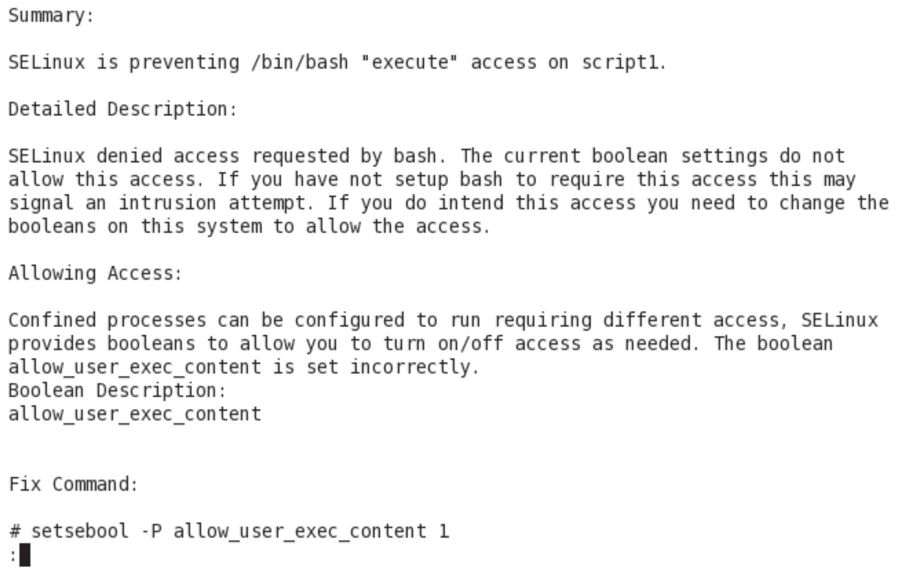


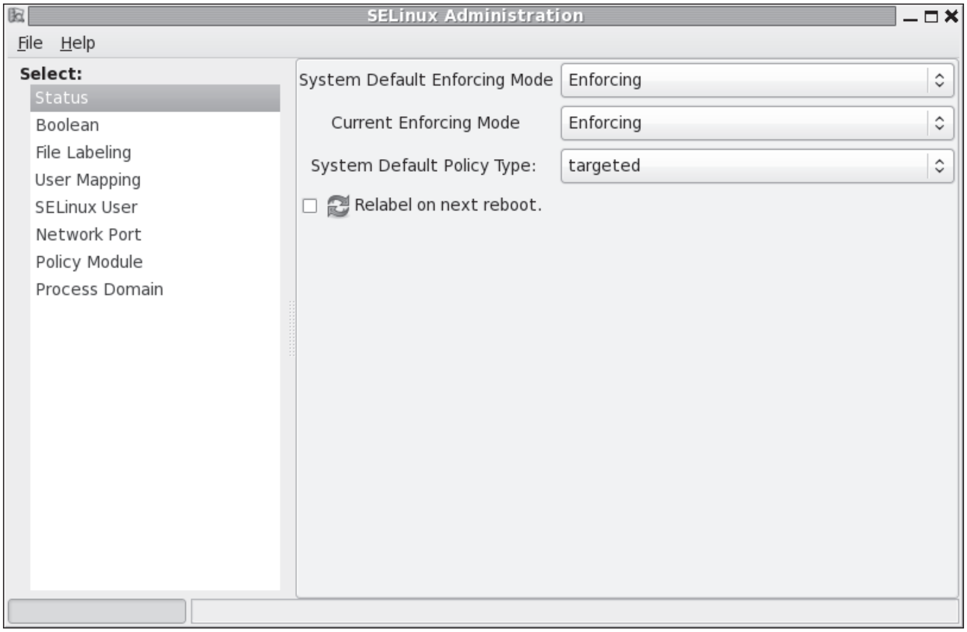
FIGURE 4-13

A SELinux alert and a solution

```
Summary:

SELinux is preventing /bin/bash "execute" access on script1.

Detailed Description:

SELinux denied access requested by bash. The current boolean settings do not
allow this access. If you have not setup bash to require this access this may
signal an intrusion attempt. If you do intend this access you need to change the
booleans on this system to allow the access.

Allowing Access:

Confined processes can be configured to run requiring different access, SELinux
provides booleans to allow you to turn on/off access as needed. The boolean
allow_user_exec_content is set incorrectly.
Boolean Description:
allow_user_exec_content


Fix Command:

# setsebool -P allow_user_exec_content 1
:
```

FIGURE 4-14

SELinux Status in the Administration tool



System Default Enforcing Mode: Enforcing

Current Enforcing Mode: Enforcing

System Default Policy Type: targeted

□ Relabel on next reboot.

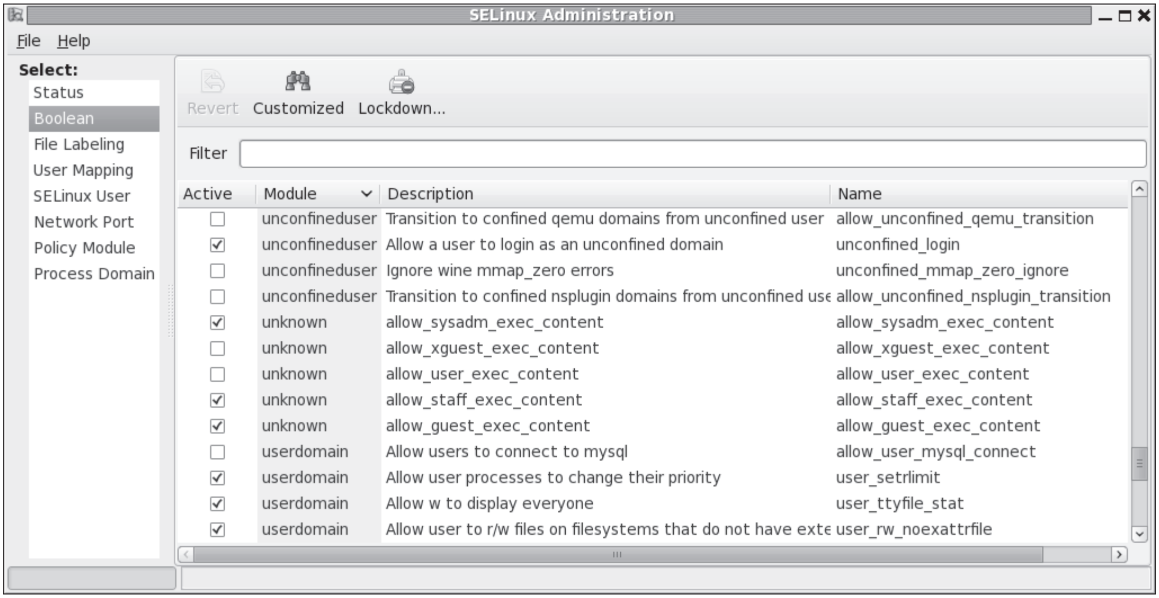
The SELinux Management tool is much more capable than the previous utility that was part of the Security Level Configuration tool. As you can see, there are options for Default and Current Enforcing Modes, which you can set to Enforcing, Permissive, or Disabled. While the focus of SELinux is on a Targeted polity, MLS is also available. Generally, you don't need to activate the Relabel On Next Reboot option unless you've changed the default policy type.

There are a number of categories shown in the left pane of the SELinux Management Tool window described in the following sections. In the RHCE half of this book, you'll revisit this tool with more of a focus on Boolean settings in the second half of this book.

SELinux Boolean Settings

In the SELinux Administration tool, click Boolean in the left-hand pane. Scroll through available modules. As you can see, SELinux policy can be modified in a number of different categories, some related to administrative functions, others to specific services. A select number of these options are shown in Figure 4-15. Any changes you make are reflected in boolean variables in the /selinux/booleans

FIGURE 4-15 Booleans in the SELinux Management tool



Active	Module	Description	Name
□	unconfineduser	Transition to confined qemu domains from unconfined user	allow_unconfined_qemu_transition
☑	unconfineduser	Allow a user to login as an unconfined domain	unconfined_login
□	unconfineduser	Ignore wine mmap_zero errors	unconfined_mmap_zero_ignore
□	unconfineduser	Transition to confined nsplugin domains from unconfined use	allow_unconfined_nsplugin_transition
☑	unknown	allow_sysadm_exec_content	allow_sysadm_exec_content
□	unknown	allow_xguest_exec_content	allow_xguest_exec_content
□	unknown	allow_user_exec_content	allow_user_exec_content
☑	unknown	allow_staff_exec_content	allow_staff_exec_content
☑	unknown	allow_guest_exec_content	allow_guest_exec_content
□	userdomain	Allow users to connect to mysql	allow_user_mysql_connect
☑	userdomain	Allow user processes to change their priority	user_setrlimit
☑	userdomain	Allow w to display everyone	user_ttyfile_stat
☑	userdomain	Allow user to r/w files on filesystems that do not have exte	user_rw_noexattrfile

directory. Module categories of interest for the RHCSA exam include cron, init, mount, qemu, and that catch-all category: unknown. As the list is relatively short, the associated booleans are listed in Table 4-9. The booleans appear in the order shown in the SELinux Management tool.

File Labeling

You can change the default labels associated with files, some of which are described earlier in this chapter (and in other chapters discussing SELinux contexts). Some of the options are shown in Figure 4-16. Any changes to this screen are written to the file_contexts.local file in the /etc/selinux/targeted/contexts/files directory.

User Mapping

The User Mapping section allows you to go beyond the defaults for regular and administrative users. The display here illustrates the current output to the **semanage login -l** command. If you don't remember the intricacies of the **semanage** command, it may be easier to use this screen to map existing users to different maps. Click Add to open the Add User Mapping window shown in Figure 4-17. That figure also

TABLE 4-9 Selected SELinux Boolean Options

Boolean	Description
fcron_crond	Supports fcron rules for job scheduling
cron_can_relabel	Allows cron jobs to change the SELinux file context label
allow_daemons_use_tty	Lets service daemons use terminals as needed
allow_daemons_dump_core	Supports writing of core files to the top-level root directory
init_upstart	Allows supplanting of SysVInit with upstart
allow_mount_anyfile	Permits the use of the **mount** command on any file
qemu_use_nfs	Supports the use of NFS filesystems for virtual machines
qemu_use_usb	Supports the use of USB devices for virtual machines
qemu_full_network	Supports networking for virtual machines
qemu_use_cifs	Supports the use of CIFS (Common Internet File System) filesystems for virtual machines
qemu_use_comm	Supports a connection for virtual machines to serial and parallel ports
allow_sysadm_exec_content	Allows sysadm_u users the right to execute scripts
allow_xguest_exec_content	Allows xguest_u users the right to execute scripts
allow_user_exec_content	Allows user_u users the right to execute scripts
allow_staff_exec_content	Allows staff_u users the right to execute scripts
allow_guest_exec_content	Allows guest_u users the right to execute scripts

illustrates how you might reclassify a user named michael as a SELinux user_u user type.

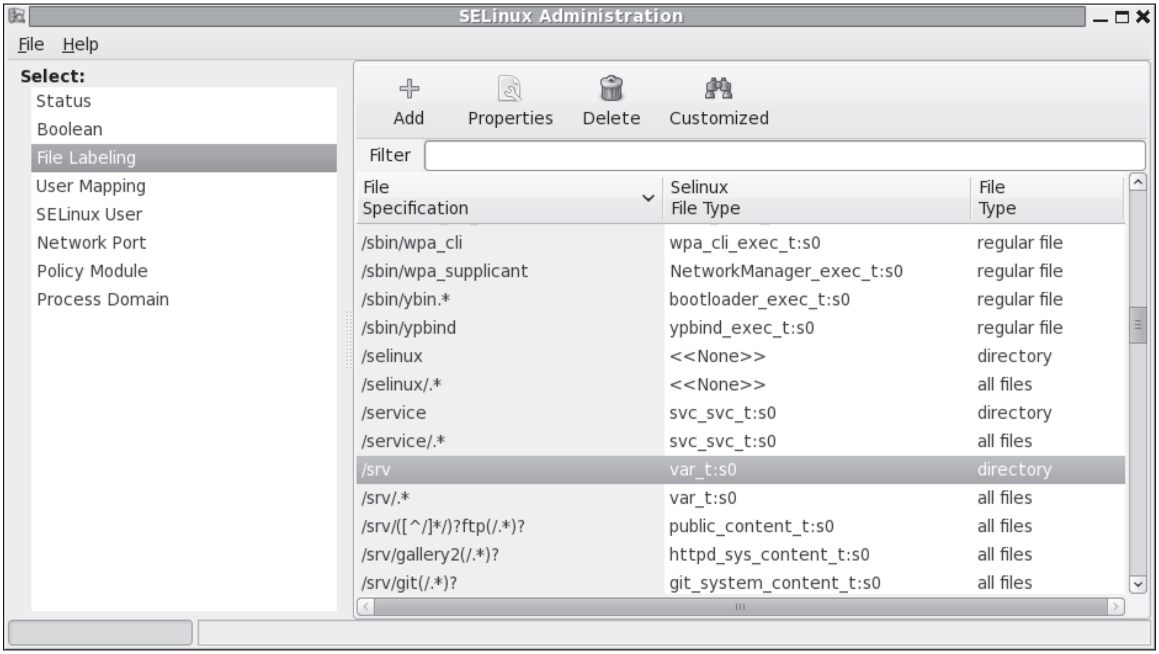
SELinux User

The SELinux User section allows you to specify and modify default roles for standard users, such as regular users (user_u), system users (system_u), unconfined users (unconfined_u), and the administrative root user.

Network Port

The Network Port section associates standard ports with services, which will be important in the RHCE section of this book.

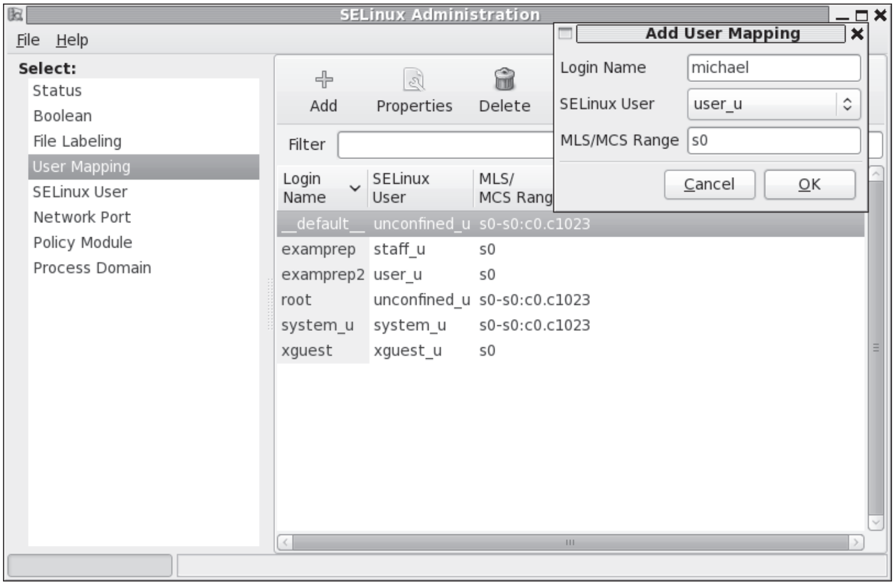
FIGURE 4-16 File types in the SELinux Management tool.



File Specification	Selinux File Type	File Type
/sbin/wpa_cli	wpa_cli_exec_t:s0	regular file
/sbin/wpa_supplicant	NetworkManager_exec_t:s0	regular file
/sbin/ybin.*	bootloader_exec_t:s0	regular file
/sbin/ypbind	ypbind_exec_t:s0	regular file
/selinux	<<None>>	directory
/selinux/.*	<<None>>	all files
/service	svc_svc_t:s0	directory
/service/.*	svc_svc_t:s0	all files
/srv	var_t:s0	directory
/srv/.*	var_t:s0	all files
/srv/([^/]*/)?ftp(/.*)?	public_content_t:s0	all files
/srv/gallery2(/.*)?	httpd_sys_content_t:s0	all files
/srv/git(/.*)?	git_system_content_t:s0	all files

FIGURE 4-17

Map a user in the SELinux Management tool.



Login Name: michael

SELinux User: user_u

MLS/MCS Range: s0

Login Name	SELinux User	MLS/ MCS Rang
__default__	unconfined_u	s0-s0:c0.c1023
examprep	staff_u	s0
examprep2	user_u	s0
root	unconfined_u	s0-s0:c0.c1023
system_u	system_u	s0-s0:c0.c1023
xguest	xguest_u	s0

Policy Module

The Policy Module section specifies the SELinux policy version number applied to each module.

Process Domain

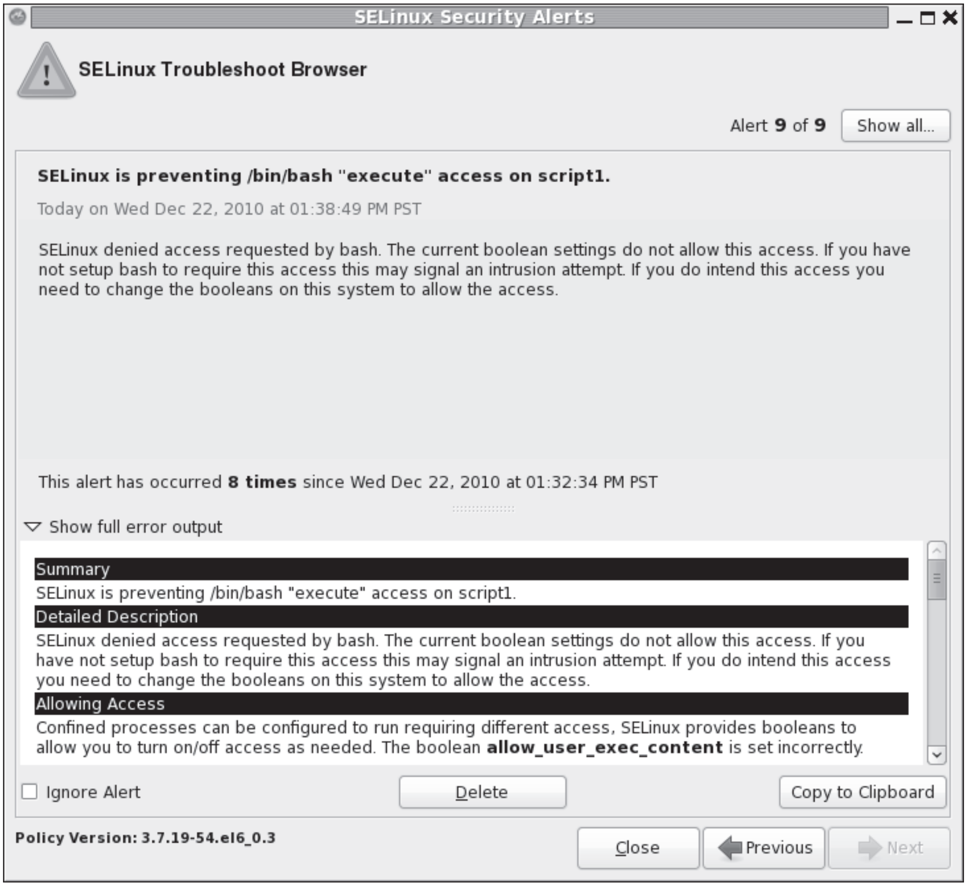
The Process Domain allows you to change the status of SELinux to Permissive or Enforcing mode.

The SELinux Troubleshoot Browser

RHEL 6 includes the SELinux Troubleshoot Browser shown in Figure 4-18. It provides tips and advice on any problems that you may encounter, in a language

FIGURE 4-18

Security alerts with the SELinux Troubleshoot Browser



SELinux Security Alerts

SELinux Troubleshoot Browser

Alert **9** of **9**

SELinux is preventing /bin/bash "execute" access on script1.

Today on Wed Dec 22, 2010 at 01:38:49 PM PST

SELinux denied access requested by bash. The current boolean settings do not allow this access. If you have not setup bash to require this access this may signal an intrusion attempt. If you do intend this access you need to change the booleans on this system to allow the access.

This alert has occurred **8 times** since Wed Dec 22, 2010 at 01:32:34 PM PST

▽ Show full error output

```
Summary
SELinux is preventing /bin/bash "execute" access on script1.
Detailed Description
SELinux denied access requested by bash. The current boolean settings do not allow this access. If you
have not setup bash to require this access this may signal an intrusion attempt. If you do intend this access
you need to change the booleans on this system to allow the access.
Allowing Access
Confined processes can be configured to run requiring different access, SELinux provides booleans to
allow you to turn on/off access as needed. The boolean allow_user_exec_content is set incorrectly.
```

□ Ignore Alert

Policy Version: 3.7.19-54.el6_0.3

more Linux administrators can understand, often including commands that you can run and that will address the subject problem.

To start the Browser from the GNOME desktop, click Applications | System Tools | SELinux Troubleshooter or run **sealert -b** from a GUI-based command line. The command is available from the setroubleshoot-server package.

EXERCISE 4-3

Test an SELinux User Type

In this exercise, you'll configure a user with the staff_u SELinux user type and test the results. You'll need a GUI, and at least one regular user other than the root administrative user.

1. If necessary, create a regular user. Even if you already have a regular user, a second regular user for the purpose of this exercise may reduce risks. Users can always be deleted, as discussed in Chapter 8. To that end, the **useradd user1** and **passwd user1** commands create a user named user1 with a password.
2. Review the SELinux types of current users with the **semanage login -l** command.
3. Configure the desired user as a staff_u user with the **semanage login -a -s staff_u user1** command. Substitute as desired for user1.
4. If you're completely logged in to the GUI, log out. Click System | Log Out, and click Log Out in the window that appears.
5. Log into the GUI with the newly revised staff_u account, user1 (or whatever else you may have configured in step 3). If you don't already see a GUI login screen, press ALT-F1 or ALT-F7.
6. Try various administrative commands. Do you have access to the **su** command? What of **sudo**? What administrative tools discussed so far in this book are accessible? Is there a difference whether that tool is started from the GUI command line or from the GUI menu?
7. Log out of the new staff_u account, and log back into the regular account,
8. Delete the new user from the staff_u list; if that's user1, you can do so with the **semanage login -d user1** command.
9. Confirm the restored configuration with the **semanage login -l** command.

SCENARIO & SOLUTION

A file can't be read, written to, or executed	Convert it before you upgrade. NT 4 does not like HPFS.
Access to a secure file required for a single user	Configure ACLs for the appropriate filesystem and then apply the **setfacl** command to provide access.
The SSH service is not accessible on a server	Assuming the SSH service is running (a RHCE requirement), make sure the firewall supports SSH access with the **iptables -L** command or the system-config-firewall tool.
Enforcing mode is not set for SELinux	Set enforcing mode with the **setenforce enforcing** command.
Need to restore SELinux default file contexts on a directory	Apply the **restorecon -F** command to the target directory.
Unexpected failure when SELinux is set in enforcing mode	Use the **sealert -a /var/log/audit/audit.log** command or the SELinux Troubleshooter to find more information about the failure; sometimes a suggested solution is included.
Need to change SELinux options for a user	Apply the **setsebool -P** command to the appropriate boolean setting.

CERTIFICATION SUMMARY

This chapter focuses on the basics of RHCSA-level security. On any Linux system, security starts with the ownership and permissions associated with a file. Ownership may be divided into users, groups, and others. Permissions may be divided into read, write, and execute, in a scheme known as discretionary access controls. Default file permissions are based on the value of **umask** for a user. Permissions may be extended with the SUID, SGID, and sticky bits.

ACLs can add another dimension to discretionary access controls. When configured on a mounted volume, ACLs can be configured to supersede basic ugo/rwx permissions. ACLs can be included in NFSv4 shared directories.

Firewalls can prevent communication on all but desired ports. Standard ports for most services are defined in the /etc/services file. However, some services may not use all of the protocols defined in that file. The default RHEL 6 firewall supports access only to a local SSH server.

SELinux provides another layer of protection, using mandatory access control. With a variety of available SELinux users, objects, file types, and MLS ranges, SELinux controls can help ensure that a breach in one service doesn't lead to trouble with other services.

TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 4.

Basic File Permissions

□ Standard Linux file permissions are read, write, and execute, which may vary for the user owner, the group owner, and other users.

□ Special permissions include the SUID, SGID, and sticky bits.

□ Default user permissions are based on the value of the **umask**.

□ Ownership and permissions can be changed with the **chown**, **chgrp**, and **chmod** commands.

□ Special file attributes can be listed with the **lsattr** and modified by the **chattr** command.

Access Control Lists and More

□ ACLs can be listed and modified on filesystems mounted with the **acl** option.

□ Every file already has ACLs based on standard ownership and permissions.

□ You can configure ACLs on a file to supersede standard ownership and permissions for specified users and groups on selected files. Actual ACLs may depend on the mask.

□ Custom ACLs on a file are not enough; selected users and groups also need access to the directories that contain such files.

□ Just as custom ACLs can support special access for selected users, it can also deny access to other selected users.

□ ACLs can be configured on shared NFS directories.

Basic Firewall Control

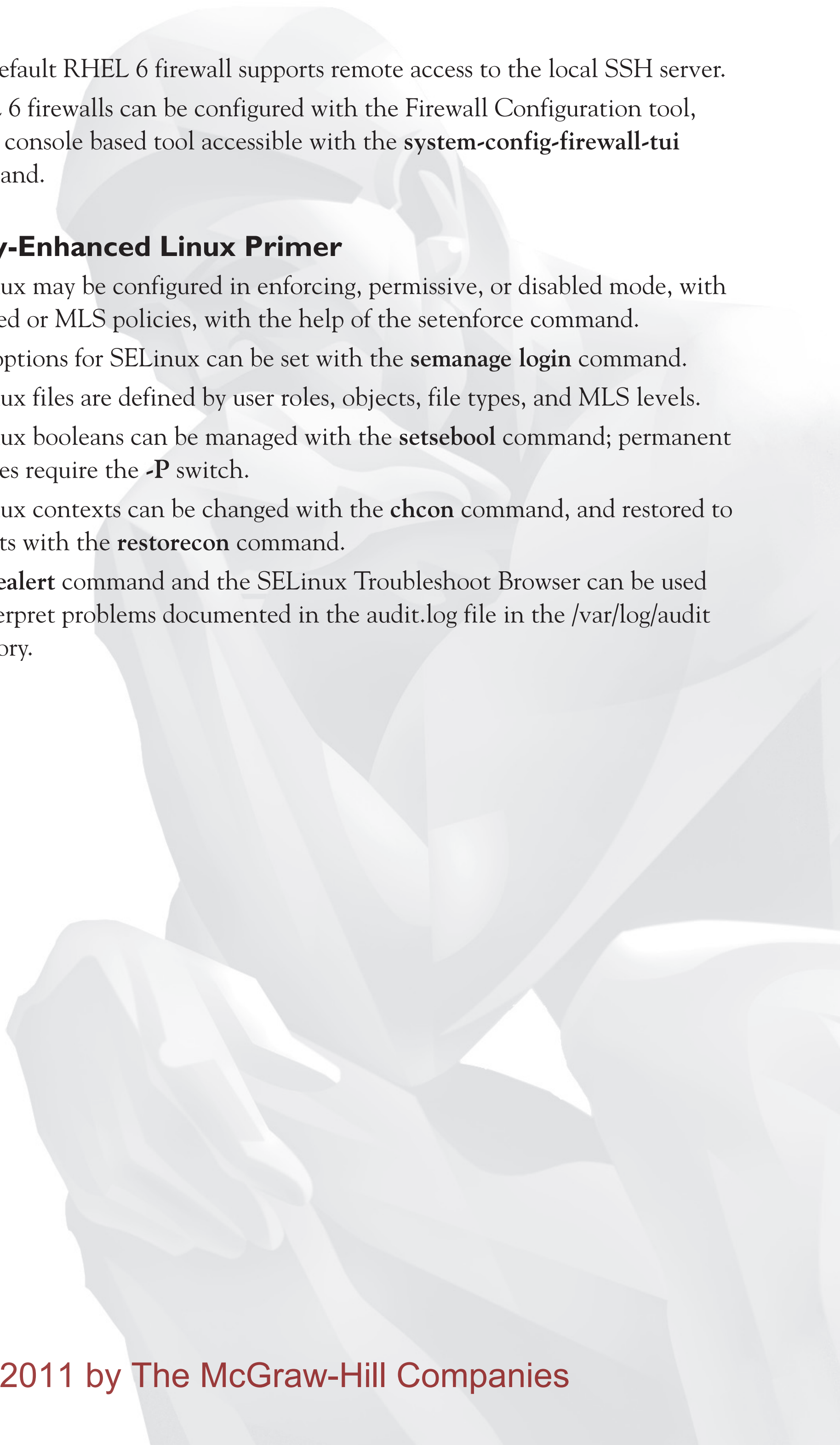
□ Standard Linux firewalls are based on the **iptables** command, with options stored in /etc/sysconfig/iptables.

□ Standard Linux firewalls assume the use of some of the ports and protocols listed in /etc/services.

- ❑ The default RHEL 6 firewall supports remote access to the local SSH server.
- ❑ RHEL 6 firewalls can be configured with the Firewall Configuration tool, or the console based tool accessible with the **system-config-firewall-tui** command.

A Security-Enhanced Linux Primer

- ❑ SELinux may be configured in enforcing, permissive, or disabled mode, with targeted or MLS policies, with the help of the setenforce command.
- ❑ User options for SELinux can be set with the **semanage login** command.
- ❑ SELinux files are defined by user roles, objects, file types, and MLS levels.
- ❑ SELinux booleans can be managed with the **setsebool** command; permanent changes require the **-P** switch.
- ❑ SELinux contexts can be changed with the **chcon** command, and restored to defaults with the **restorecon** command.
- ❑ The **sealert** command and the SELinux Troubleshoot Browser can be used to interpret problems documented in the audit.log file in the /var/log/audit directory.



SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. As no multiple-choice questions appear on the Red Hat exams, no multiple-choice questions appear in this book. These questions exclusively test your understanding of the chapter. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one right answer to many of these questions.

Basic File Permissions

1. What command configures read and write permissions on the file named question1 in the local directory, with no permissions for any other user?

2. What single command changes the user owner to professor and group owner to assistants for the local file named question2?

3. What command would change the attributes of a file named question3 to allow you to only add to that file?

Access Control Lists and More

4. What command would add ACLs to the mount of the /dev/sda2 partition on the /home directory? Assume the filesystem is already mounted.

5. What command reads current ACLs for the local file named question5? Assume that file is on a filesystem mounted with ACLs.

6. What single command gives members of the group named managers read access to the project6 file in the /home/project directory? Assume the managers group already has read and execute access to the directory.

7. What command prevents members of the group named temps from having any access to the secret7 file in the /home/project directory?

____

Basic Firewall Control

8. What TCP/IP port number is associated with the FTP service?

____

9. Name the full path to the file with RHEL 6 firewall configuration rules based on the **iptables** command.

____

A Security-Enhanced Linux Primer

10. What command configures SELinux in enforcing mode?

____

11. What command lists the SELinux status of current users?

____

12. What directory includes boolean settings for SELinux?

____

LAB QUESTIONS

Several of these labs involve configuration exercises. You should do these exercises on test machines only. It's assumed that you're running these exercises on virtual machines such as KVM, and they're not used for production.

Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter4/ subdirectory. In case you haven't yet set up RHEL 6 on a system, refer to Chapter 1 for installation instructions.

The answers for each lab follows the self test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

Basic File Permissions

1. The command that configures read and write permissions on the file named question1 in the local directory, with no permissions for any other user is:

```
# chmod 600 question1
```

2. The single command that changes the user owner to professor and group owner to assistants for the noted file is:

```
# chown professor.assistants question2
```

   It's acceptable to substitute a colon (:) for the dot (.).

3. The command that change the attributes of a file named question3 to allow you to only add to that file is:

```
# chattr +a question3
```

Access Control Lists and More

4. The command that adds ACLs to the mount of the /dev/sda2 partition on the /home directory is:

```
# mount -o remount,acl /dev/sda2 /home
```

   Since you don't know whether the filesystem is configured in /etc/fstab, you need the device a nd directory for the command.

5. The command that reads current ACLs for the local file named question5 is:

```
# getfacl question5
```

6. The single command gives that members of the group named managers read access to the project6 file in the /home/project directory is:

```
# setacl -m g:managers:r /home/project/project6
```

7. The command that prevents members of the group named temps from having any access to the secret7 file in the /home/project directory is:

```
# setacl -m g:temps:--- /home/project/secret7
```

Basic Firewall Control

8. The TCP/IP port number associated with the FTP service is 21.
9. The full path to the file with RHEL 6 firewall configuration rules is /etc/sysconfig/iptables.

A Security-Enhanced Linux Primer

10. The command that configures SELinux in enforcing mode is:

    ```
    # setenforce enforcing
    ```

11. The command that lists the SELinux status of current users is:

    ```
    # semanage login -l
    ```

12. The directory includes boolean settings for SELinux is:

    ```
    /selinux/booleans
    ```

LAB ANSWERS

Lab 1

In some ways, Lab 1 could have been split into two different labs. It's designed to let you practice configuring those permissions required to set up executable scripts for user owners and other users. Success with the script is straightforward; if it's executed, you'll find a file named filelist, with a list of files, in the local directory.

It's also designed to help you understand the effect of the SUID bit of /usr/bin/passwd, a compiled executable.

Lab 2

Lab 2 is essentially an extension of Lab 1, in that ACLs are another approach to making a script, owned by the root administrative user, executable by a single regular user. You can check for success in the same way; if the script is properly executed by the ACL configured regular user, you'll find a file named filelist in the local directory.

Lab 3

The configuration of ACLs on the /root administrative directory is a bad security practice. However, it is an excellent way to illustrate the capabilities of ACLs on a system, how it can allow access by selected regular users to the inner sanctums of the root administrative account. Because of the risks, just be sure to disable the ACLs when the lab is complete. If the selected user is michael, one method is with the following command:

```
# setfacl -b u:michael /root
```

If you've configured ACLs on files within the /root directory, be sure to disable those as well. Of course, in standard Linux configurations, the /root directory is mounted on the same volume as the top-level root directory (/). During the lab, you should have mounted that top-level root directory with the acl option. That should be confirmed in the output to the **mount** command. Given the risks, you should restore the original configuration with the following command:

```
# mount -o remount /
```

Lab 4

This lab is designed to raise awareness of the time and effort required to disable and re-enable SELinux in enforcing mode. If you switch between disabled and permissive mode, the time and effort required should be about the same. While it may not seem to take that much time, if you have to reconfigure SELinux in enforcing mode, it might seem to take "forever" as nothing else can be done while the system is being rebooted and relabeled.

Lab 5

Standard users in RHEL 6 run as unconfined_u SELinux user types. As such, there are few limits on their user accounts. If instructions on an exam or from a corporate policy require certain limits on regular users, you may want to set up the __default__ user with the SELinux user_u user type. Alternatively, if you're told to set up specific users to a limited type, such as xguest_u or staff_u, multiple **semanage login** commands may be appropriate.

Lab 6

After testing a user as a guest_u user, most administrators will want regular users to have more privileges.

Lab 7

Users configured with the guest_u SELinux user type are not normally allowed to execute scripts even in their own home directories. That can change with the allow_guest_exec_content boolean described in the lab. Success in this lab is based on a simple comparison; whether a script can be executed with and without the active boolean.

While the easiest way to restore the original configuration is with the GUI SELinux management tool, you should also know how to use commands like the following, which disables a custom SELinux user type for user michael:

```
# semanage login -d michael
```

Lab 8

Success in this lab can be measured first with the **ls -Zd** command. When applied to both the /ftp and the /var/ftp/pub directories, it should lead to the same list of SELinux roles, objects, types, and MLS options for each directory.

This lab also contains a solvable mystery. Why is it that the SELinux contexts of a newly created directory differ from those where the contexts have been restored with the **restorecon** command.

The basic reason is the difference between regular unconfined_u SELinux user types and the options listed in the file_contexts file in the /etc/selinux/targeted/contexts/files directory. You'll learn how to modify default file contexts in Chapter 11.

Lab 9

Everyone will experiment with SELinux in different ways. So the results of this lab are up to you. The objective is to analyze a current relevant log file, and process it at the command line. Try to identify the problems associated with each alert. While you may not be able to address many SELinux issues, at least until the second half of this book, you should be able to identify the problems or at least the users and/or commands associated with each alert. If you don't have enough of a log, two related files are available in the Chapter4/ directory of the CD.