Chapter 8

User Administration

CERTIFICATION OBJECTIVES

8.01	User Account Management	8.05	Special Groups
8.02	Administrative Control	✓	Two-Minute Drill
8.03	User and Shell Configuration	Q&A	Self Test
8.04	Users and Network Authentication		

Fundamental to Linux administration is the management of users and groups. In this chapter, you'll examine different ways to manage the variety of users and groups available to Linux. Important skills in this area range from the simple login to user account management, group membership, group collaboration, and network authentication. The configuration of administrative privileges for Linux users can help the master administrator distribute responsibilities to others.

You'll see how to manage these tasks from the command line, with the help of the files of the shadow password suite. You'll also use tools such as the User Manager and the Authentication Configuration tool to set up some of these tasks. As you should expect, Red Hat GUI tools can't do it all, which emphasizes the importance of understanding user management from the command line.

INSIDE THE EXAM

Inside the Exam

This chapter addresses several RHCSA objectives. Briefly explained, these objectives include the following:

- Log in and switch users in multiuser targets

Briefly, that means you need to know how to log in with regular accounts when RHEL 7 is running in the multiuser or graphical targets. To switch users, you need to know how to log out and log back in with a second account. Simple enough.

- Create, delete, and modify local user accounts
- Change passwords and adjust password aging for local user accounts
- Create, delete, and modify local groups and group memberships

You could use commands such as **useradd**, **usermod**, **groupadd**, **groupmod**, and **chage** as well as the User Manager to accomplish these tasks. While this chapter explains how you can use both types of tools, there is no guarantee that the User Manager will be available during an exam.

- Create and configure set-GID directories for collaboration

When the RHCT exam was available, the related objective was "Configure filesystem permissions for collaboration." In other words, the objective is now more specific—you're told how to set one or more directories for collaboration between a group of users.

- Configure a system to use an existing authentication service for user and group information

In the previous version of the RHCSA exam for RHEL 6, this objective was limited to "Attach a system to a centralized Lightweight Directory Access Protocol (LDAP) server." Now the scope of the objective is broader, and may include any service such as Kerberos, Microsoft Active Directory, or an Identity, Policy, and Audit (IPA) server.

CERTIFICATION OBJECTIVE 8.01

User Account Management

You need to know how to create and configure users. This means knowing how to configure and modify accounts, work with passwords, and organize users in groups. You also need to know how to configure the environment associated with each user account: in configuration files and in user settings.

If you've installed RHEL 7 via Kickstart or in text mode, or otherwise avoided the Firstboot process described in Chapter 1, the default Red Hat installation includes just a single login account: root. Although no other accounts are required, it's important to set up some regular user accounts. Even if you're going to be the only user on the system, do create at least one non-administrative account for day-to-day work. Then you can use the root account only when it's necessary to administer the system. You can add accounts to Red Hat Enterprise Linux systems using various utilities, including direct editing of password configuration files (the manual method), the **useradd** command (the command-line method), and the User Manager utility (the graphical method).

Different Kinds of Users

There are three basic types of Linux user accounts: administrative (root), regular, and service. The administrative root account is automatically created when Linux is installed, and it has administrative privileges for all services on a Linux system. A "black hat" hacker who has a chance to take control of this account can take full control of that system.

For the times when you do log in as an administrator, RHEL builds in safeguards for root users. Log in as the root user, and then run the **alias** command. You'll see entries such as the following:

```
alias rm='rm -i'
```

Due to this particular alias, when the root user runs the **rm** command, the shell actually executes the **rm -i** command, which prompts for confirmation before the **rm** command deletes a file. Unfortunately, a command such as **rm -rf *directoryname*** supersedes this safety setting.

Regular users have the necessary privileges to perform standard tasks on a Linux computer. They can access programs such as word processors, databases, and web browsers. They can store files in their own home directories. Since regular users do not normally have administrative privileges, they cannot accidentally delete critical operating system configuration files. You can assign a regular account to most users, safe in the knowledge that they can't disrupt a system with the privileges they have on that account.

Services such as Apache, Squid, mail, and printing have their own individual service accounts. These accounts exist to allow each of these services to interact with Linux systems. Normally, you won't need to change any service account, but if you see that someone is running a Bash shell through one of these accounts, be wary. Someone may have broken into your system.

on the job

To review recent logins, run the last | less **command. If the login is from a remote location, it will be associated with a specific IP address outside your network.**

The Shadow Password Suite

When Unix was first developed back in the 1970s, security was not a serious concern. Everything required for user and group management was contained in the /etc/passwd and /etc/group files. As suggested by the name, passwords were originally in the /etc/passwd file. The problem is that file is "world readable." Anyone with a copy of that file, before the shadow password suite, would have a copy of the password for every user. Even passwords that are encrypted in that file may eventually be decrypted. That is the motivation behind the development of the shadow password suite, where more sensitive information was moved to other files, readable only by the root administrative user.

The four files of the shadow password suite are /etc/passwd, /etc/group, /etc/shadow, and /etc/gshadow. Defaults in these files are driven by the /etc/login.defs file.

The /etc/passwd File

The /etc/passwd file contains basic information about every user. Open that file in a text editor and browse around a bit. At the top of the file is basic information for the root administrative user. Other users in this file may relate to services such as mail, ftp, and sshd. They may be specific users designed for logins.

There are seven columns of information in the /etc/passwd file, delineated by colons. Each column in /etc/passwd includes specific information described in Table 8-1.

The RHEL 7 version of /etc/passwd includes more secure features for user accounts when compared to some other Linux distributions. The only accounts with a real login shell are user accounts. If a "black hat" hacker somehow breaks into a service account, such as mail or nobody, with the false /sbin/nologin shell, that user doesn't automatically get access to the command line.

The /etc/group File

Every Linux user is assigned to a group. By default, in RHEL 7 every user gets his own private group. The user is the only member of that group, as defined in the /etc/group configuration file. Open that file in a text editor. Browse around a bit. The first line in this

TABLE 8-1 The Anatomy of /etc/passwd

Field	Example	Purpose
Username	mj	The user logs in with this name. Usernames can include digits, hyphens (-), dots (.), and underscores (_). However, they should not start with a hyphen or be longer than 32 characters.
Password	x	The password. You should see either an *x*, an asterisk (*), or a seemingly random group of letters and numbers. An *x* points to /etc/shadow for the actual password. An asterisk means the account is disabled. A random group of letters and numbers represents the encrypted password.
User ID	1000	The unique numeric user ID (UID) for that user. By default, Red Hat starts user IDs at 1000.
Group ID	1000	The primary group ID (GID) associated with that user. By default, RHEL creates a new group for every new user, and the number matches the UID, if the corresponding GID is available. Some other Linux and Unix systems assign all users to a default Users group.
User info	Michael Jang	You can enter any information of your choice in this field. Standard options include the user's full name, telephone number, e-mail address, and physical location. You can leave this blank.
Home Directory	/home/mj	By default, RHEL places new home directories in /home/*username.*
Login Shell	/bin/bash	By default, RHEL assigns users to the bash shell. You can change this to any legal shell you have installed.

file specifies information for the root administrative user's group. Some service users include other users as members of that group. For example, user qemu is a member of the kvm group, which gives services associated with the QEMU emulator privileges with the Kernel-based Virtual Machine (KVM).

There are four columns of information in the /etc/group file, delineated by colons. Each column in the /etc/group specifies information described in Table 8-2.

The /etc/shadow File

The /etc/shadow file is a supplement to /etc/passwd. It contains eight columns of information, and the first column contains the same list of usernames as documented in /etc/passwd. As long as there's an *x* in the second column of each /etc/passwd entry, Linux knows to look at /etc/shadow for more information. Open that file in a text editor and browse around a bit. You'll see the same pattern of information, starting with the root administrative user.

TABLE 8-2 The Anatomy of /etc/group

Field	Example	Purpose
Groupname	mj	Each user gets his own group, with the same name as his username. You can also create unique group names.
Password	x	The password. You should see either an *x* or a seemingly random group of letters and numbers. An *x* points to /etc/gshadow for the actual password. A random group of letters and numbers represents the encrypted password.
Group ID	1000	The numeric group ID (GID) associated with the group. By default, RHEL creates a new group for every new user. If you want to create a special group such as managers, you should assign a GID number outside the standard range; otherwise, Red Hat GIDs and UIDs would probably get out of sequence.
Group members	mj,vp,ao	Lists the usernames that are members of the group. If there is a username that lists the GID of the group as its primary group in /etc/passwd, that username is also a member of the group.

As shown in Table 8-3, /etc/shadow includes the encrypted password in the second column, and the remaining information relates to the way passwords are managed. In fact, the first two characters of the second column are based on the encryption hash for the password. If you see a $1, the password is hashed to the Message Digest 5 (MD5) algorithm, the standard through RHEL 5. If you see a $6, the password is protected with the 512-bit Secure Hash Algorithm (SHA-512), the standard for RHEL 6 and 7.

TABLE 8-3 The Anatomy of /etc/shadow

Column	Field	Description
1	Username	Username
2	Password	Encrypted password; requires an *x* in the second column of /etc/passwd
3	Password history	Date of the last password change, in number of days after January 1, 1970
4	mindays	Minimum number of days that a user must keep a password
5	maxdays	Maximum number of days after which a password must be changed
6	warndays	Number of days before password expiration when a warning is given
7	inactive	Number of days after password expiration during which the password is still accepted, but the user is prompted to change her password
8	disabled	Number of days since January 1, 1970, after which an account is disabled

TABLE 8-4 The Anatomy of /etc/gshadow

Field	Example	Purpose
Groupname	mj	The group name.
Password	!	Most groups have a !, which indicates no password; some groups may have a hashed password similar to that shown in the /etc/shadow file.
Administrators	mj	A comma-separated list of users who are part of the group and can change the members or the password of the group using the **gpasswd** command.
Group members	vp,ao	A comma-delineated list of usernames that are members of the group.

The /etc/gshadow File

The /etc/gshadow file is the group configuration file in the shadow password suite. It includes the group administrators, which can add other group members using the **gpasswd** command. If desired, you can even configure a hashed password. Once a password is set, other users can become members of the group by using the **newgrp** command and typing the required password. Table 8-4 describes the columns in /etc/gshadow, from left to right.

The /etc/login.defs File

The /etc/login.defs file provides the baseline for a number of parameters in the shadow password suite. This section provides a brief analysis of each active directive in the default version of this file. As you'll see, the directives go somewhat beyond authentication. The first configuration parameter specifies the directory with locally delivered e-mail, listed by username:

```
MAIL_DIR /var/spool/mail
```

The next four directives relate to default password aging information. The directives are explained in the file comment and in Table 8-5.

TABLE 8-5 /etc/login.defs Password-Aging Configuration Parameters

Configuration Parameter	Purpose
PASS_MAX_DAYS	After this number of days, the password must be changed.
PASS_MIN_DAYS	Passwords must be kept for at least this number of days.
PASS_MIN_LEN	The length of a password must be at least this number of characters.
PASS_WARN_AGE	Users are warned this number of days before PASS_MAX_DAYS.

As suggested earlier, User ID (UID) and Group ID (GID) numbers for regular users and groups start at 1000. Since Linux supports UID and GID numbers above 4 billion (actually, up to 2^{32}–1), the maximum UID and GID numbers of 60000, as defined in the /etc/login.defs file, may seem strange. However, it leaves higher numbers available for other authentication databases, such as those associated with LDAP and Microsoft Windows (via Winbind). As suggested by the directives, **UID_MIN** specifies the minimum UID, **UID_MAX** specifies the maximum UID, and so on:

```
UID_MIN  1000
UID_MAX  60000
GID_MIN  1000
GID_MAX  60000
```

Similarly, the **useradd** and **groupadd** commands with the **-r** switch create a system user or system group, respectively, whose ID is chosen within the following range:

```
SYS_UID_MIN 201
SYS_UID_MAX 999
SYS_GID_MIN 201
SYS_GID_MAX 999
```

Normally, when the **useradd** command is run to create a new user, it automatically creates home directories as well, which is confirmed by the following directive:

```
CREATE_HOME yes
```

As described later in this chapter, other files set the value of the umask. But if those other files did not exist, this directive would govern the default umask for regular users:

```
UMASK   077
```

The following directive is critical in the implementation of the User Private Group scheme, where new users are also made members of their own private group, normally with the same UID and GID numbers. It means when new users are created (or deleted), the associated group is also added (or deleted):

```
USERGROUPS_ENAB yes
```

Finally, the following directive determines the algorithm used to encrypt passwords, normally SHA 512 for RHEL 7:

```
ENCRYPT_METHOD SHA512
```

Different encryption methods may be set up with the Authentication Configuration tool described later in this chapter.

Command-Line Tools

There are two basic ways to add users through the command-line interface. You can add users directly by editing the /etc/passwd file in a text editor such as vi. To this end, both **vipw** and **vigr** were described in Chapter 3. Alternatively, you can use text commands customized for the purpose.

exam watch

The tools discussed in this section can help you create, delete, and modify local user accounts.

Add Users Directly

Open the /etc/passwd file in the text editor of your choice. As described in Chapter 3, you can do so with the **vipw** command. However, if you add users by directly editing the files of the shadow password suite, you'll have to do two more things:

- *Add a user home directory*. For example, for user donna, you'd have to add the /home/donna home directory, making sure that user donna and group donna both have ownership of that directory.
- *Populate the user home directory*. The default option is to copy the files from the /etc/skel directory, discussed later in this chapter. You'd also have to make sure that user donna and group donna have ownership of those files copied to the /home/donna directory.

Add Users to a Group Directly

Every Linux user is assigned to a group, at least his own private group. As implied in Chapter 3, the GID number listed in the /etc/group file should usually match that shown for that user in the /etc/passwd file. The user is the only member of that group.

Of course, users can be members of other groups as well. For example, to create a group named project, you could add the entries to the /etc/group and /etc/gshadow files. One way to do so in a text editor is with the **vigr** command. As an example, the following entry might be appropriate for a group named project:

```
project:x:60001:
```

The number 60001 is used, as that is beyond the limit of the **GID_MAX** directive from the /etc/login.defs file described earlier. But that's just arbitrary. There's no prohibition against a lower number, as long as it does not interfere with existing GIDs. However, it is convenient when the UID and GID numbers of regular users match.

Of course, for a group to be useful, you'd have to add users already configured in the /etc/passwd file at the end of the line. The following example assumes these users already exist:

```
project:x:60001:michael,elizabeth,stephanie,tim
```

You'd also have to add this group to the /etc/gshadow file. You could do so directly with the **vigr -s** command. Alternatively, to set up a group administrator with a password, you could run the **gpasswd** command. For example, the **gpasswd project** command would set up a password for administering the group, associated with the **newgrp** and **sg** commands described later in this chapter. It would automatically add the encrypted password with the given group name to the /etc/gshadow file.

Add Users at the Command Line

Alternatively, you can automate this process with the **useradd** command. The **useradd pm** command would add user pm to the /etc/passwd file. In addition, the **useradd** command creates the /home/pm home directory, adds the standard files from the /etc/skel directory, and assigns the default shell, /bin/bash. But **useradd** is versatile. It includes a number of command options, as shown in Table 8-6.

Assign a Password

Once a new user is created, you can use the **passwd *username*** command to assign a password to that user. For example, the **passwd pm** command prompts you to assign a new password to user pm. RHEL is configured to avoid passwords that are based on dictionary

TABLE 8-6 useradd Command Options

Option	Purpose
-u *UID*	Overrides the default assigned *UID.* By default, in RHEL this starts at 1000 and can continue sequentially to the maximum number of users supported by kernel 2.6, which is 2^{32}–1, something over four billion users.
-g *GID*	Overrides the default assigned *GID.* If available, RHEL uses the same *GID* and *UID* numbers for each user. If you assign a *GID,* it must be either 100 (users) or already otherwise exist.
-c *info*	Enters the comment of your choice about the user, such as her name.
-d *dir*	Overrides the default home directory for the user, /home/*username.*
-e *YYYY-MM-DD*	Sets an expiration date for the user account.
-f *num*	Specifies a number of days after password expiration when the account is disabled.
-G *group1*, *group2*	Makes the user a member of *group1* and *group2,* based on their current names as defined in the /etc/group file. A space between *group1* and *group2* would lead to an error.
-s *shell*	Overrides the default shell for the user, /bin/bash.

words, shorter than eight characters, too simple, based on palindromes, and other, similar criteria for security reasons. Nevertheless, such passwords are legal and accepted if the **passwd** command is run by the root user.

Add or Delete a Group at the Command Line

When it's appropriate to add a special group to the shadow password suite, you may want to use the **groupadd** command. Generally, you'll want to use it with the **-g** switch. For example, the following command would set up a special project group with a GID of 60001:

```
# groupadd -g 60001 project
```

If you don't use the **-g** switch, the **groupadd** command takes the next available GID number. For example, if two regular users are configured on a system, they each have UID and GID numbers of 1000 and 1001, respectively. If you've run the **groupadd project** command without specifying a GID number, the project group is assigned a GID of 1002. The next regular user that's created would get a UID of 1002 and a GID of 1003, which could lead to confusion.

Fortunately, the command to delete a group is simpler. If the project group has completed its work, you can delete that group from the shadow password suite database with the following command:

```
# groupdel project
```

Delete a User

The removal of a user account is a straightforward process. The easiest way to delete a user account is with the **userdel** command. By default, this command does not delete that user's home directory, so administrators can transfer files from that user perhaps to an employee who has taken over the tasks of the deleted user. Alternatively, the **userdel -r *username*** command deletes that user's home directory along with all of the files stored in that home directory.

This is a lot faster than the GUI method, for which you open the Red Hat User Manager, select the user, and then click Delete. Although it's probably easier for a less experienced user to remember the GUI method, text commands are faster.

exam watch

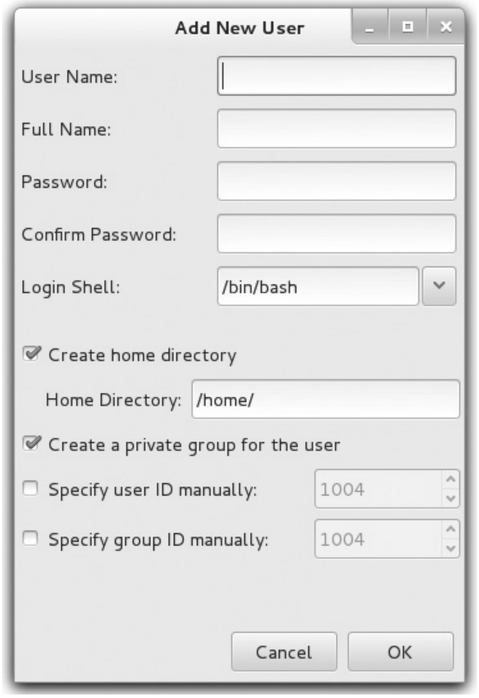
If you know both the text and GUI methods to perform a task, use the text method. It almost always saves time.

EXERCISE 8-1

Add a User with the Red Hat User Manager

If the GUI is available, one alternative to user management commands such as **useradd** and **usermod** is the Red Hat User Manager. If possible, try to open it remotely over an **ssh -X** connection, as described in Chapter 2. For example, if you've configured the server1 .example.com system as described in earlier chapters, connect to that system from a remote GUI with the **ssh -X root@192.168.122.50** command. Once logged in, enter the **system-config-users** command.

1. In the Red Hat User Manager, click the Add User button, or choose File | Add User. This will open the Add New User window, as shown here:



User Name:
Full Name:
Password:
Confirm Password:
Login Shell: /bin/bash
✓ Create home directory
Home Directory: /home/
✓ Create a private group for the user
☐ Specify user ID manually: 1004
☐ Specify group ID manually: 1004

2. Complete the form. All entries are required, except Full Name. The entries are fairly self-explanatory (see the earlier discussions of each field). The password should be at least eight characters and should ideally contain a mix of upper- and lowercase letters, numbers, and punctuation to keep it more secure from the standard password-cracking programs.

3. Enter the identical password in the Confirm Password field.
4. Note the number associated with the Specify User ID Manually and Specify Group ID Manually options; those are the UID and GID numbers that will be assigned to the new user. Click OK when you are done.
5. Repeat the process as desired for any additional new users that may be required. Make sure to create at least one new user prior to running Exercise 8-2.

EXERCISE 8-2

Real and Fake Shells

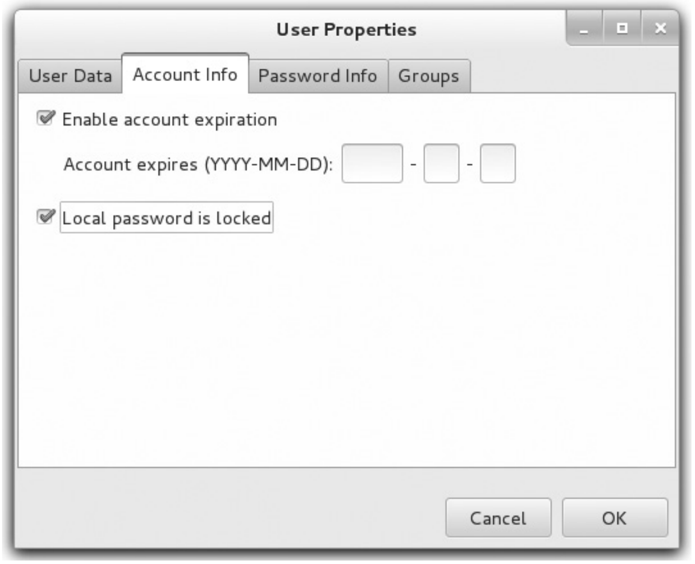
Do not run this exercise unless a regular user has already been created on the local system. If desired, run Exercise 8-1 first, as that allows you to create a new regular user on the target system.

1. Open the /etc/passwd file. Find a current regular user, with a UID of 1000 or above.
2. Identify the default shell. It's specified in the last column, normally /bin/bash for regular users.
3. Change the default shell to /sbin/nologin, and save the changes to the /etc/passwd file.
4. Open a different virtual console. Press the CTRL-ALT-F2 keys to open a different console. (If you're already in the second virtual console, substitute F3, F4, F5, or F6 for F2. If you're in a KVM-based VM in the GUI, you can move to the second virtual console by clicking Send Key | CTRL-ALT-F2.)
5. Try logging in as the modified user. What happens?
6. Return to the original console. If it's the GUI, it should be accessible with the CTRL-ALT-F1 key combination. If that is not possible (such as when the GUI is not installed), you should still be able to log in as the root administrative user.
7. Reopen the /etc/passwd file. Restore the /bin/bash shell to the target regular user.

Modify an Account

As a Linux administrator, you may want to add some limitations to user accounts. The easiest way to illustrate some of the changes is with the User Manager tool. Start the User Manager, select a currently configured user, and then click Properties to open the User Properties dialog box.

FIGURE 8-1



☑ Enable account expiration

Account expires (YYYY-MM-DD): [ ] - [ ] - [ ]

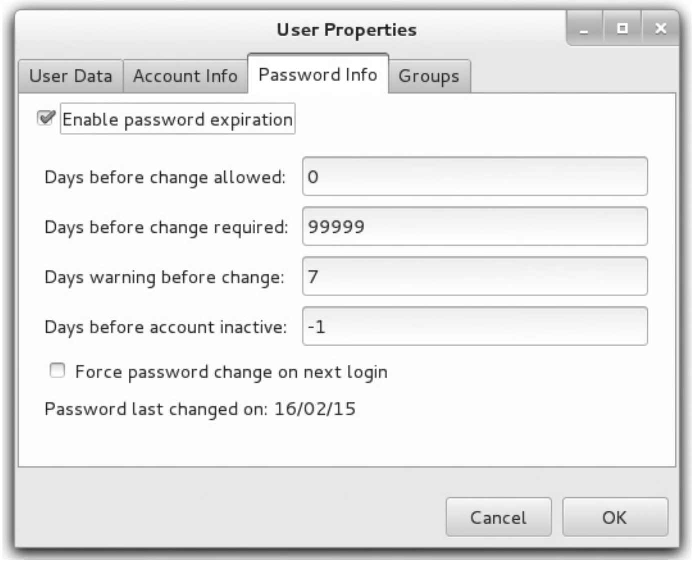
☑ Local password is locked

Manage user account life.

Click the Account Info tab for the account expiration information shown in Figure 8-1. As shown in the figure, you can limit the life of an account so that it expires on a specific date, or you can disable an account by locking it.

Click the Password Info tab. As shown in Figure 8-2, you can set several characteristics related to an individual user's password. Even when good passwords are set, frequent

FIGURE 8-2



☑ Enable password expiration

Days before change allowed:	0
Days before change required:	99999
Days warning before change:	7
Days before account inactive:	-1

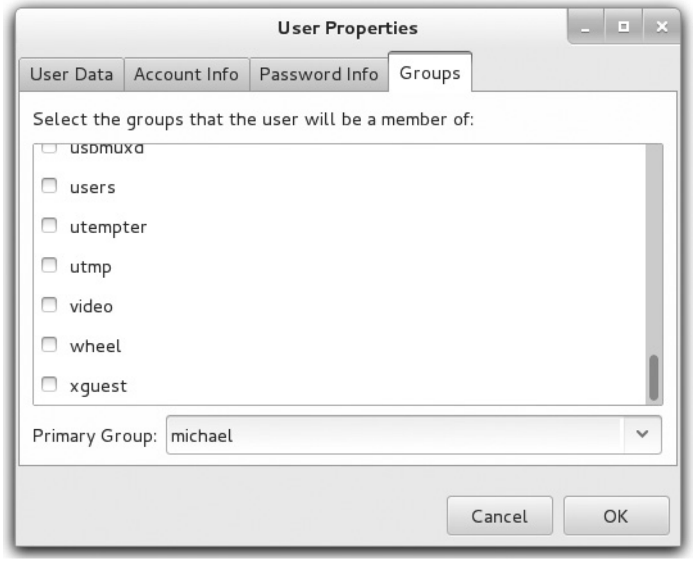
☐ Force password change on next login

Password last changed on: 16/02/15

Configure password information.

FIGURE 8-3

Assign a group.



password changes can help provide additional security. The categories shown in the figure are self-explanatory.

Click the Groups tab. Users can belong to more than one group in Linux. Under the Groups tab shown in Figure 8-3, you can assign the target user to other groups. For example, to share files and facilitate collaboration within a management team, you can assign appropriate users to a group named "managers." You can assign members of a team to the appropriate group through the Groups tab.

More User and Group Management Commands

While the Red Hat User Manager GUI utility is convenient, it's often faster to perform the associated administrative functions at the command-line interface. We've described some of these commands, such as **useradd**, **userdel**, **groupadd**, and **groupdel**. Three other key user administration commands are **usermod**, **groupmod**, and **chage**.

usermod

The **usermod** command modifies various settings in /etc/passwd. In addition, it allows you to set an expiration date for an account or an additional group. For example, the following command sets the account associated with user test1 to expire on June 8, 2016:

```
# usermod -e 2016-06-08 test1
```

TABLE 8-7

usermod Command Options

Option	Purpose
-a -G *group1*	Appends to existing group memberships; multiple groups may be specified, split with a comma, with no spaces.
-l *newlogin*	Changes the username to *newlogin,* without changing the home directory.
-L	Locks a user's password.
-U	Unlocks a user's password.

The following command makes user test1 a member of the special group:

```
# usermod -G special test1
```

The **usermod** command is closely related to the **useradd** command; in fact, the **usermod** command can use all of the **useradd** command switches listed earlier in Table 8-6. The **usermod** command includes several additional switches, as listed in Table 8-7.

groupmod

The **groupmod** command is relatively simple. It has two practical uses. The following command changes the GID number of the group named project (in this case, to 60002):

```
# groupmod -g 60002 project
```

In contrast, the following command changes the name of the group named project to secret:

```
# groupmod -n secret project
```

exam watch

The chage **command is an excellent way to address the RHCE objective to "adjust password aging for local user accounts."**

chage

The **chage** command is primarily used to manage aging information for a password, as stored in the /etc/shadow file. While some of the parameters can also be set with the **useradd** and **usermod** commands, most of the switches are different, as described in Table 8-8.

TABLE 8-8 chage Command Options

Option	Purpose
-d YYYY-MM-DD	Sets the last change date for a password; output shown in /etc/shadow as the number of days after January 1, 1970.
-E YYYY-MM-DD	Assigns the expiration date for an account; output shown in /etc/shadow as the number of days after January 1, 1970.
-I *num*	Locks an account *num* days after a password has expired; can be set to -1 to make the account permanent.
-l	Lists all aging information.
-m *num*	Sets a minimum number of days that a user must keep a password.
-M *num*	Sets a maximum number of days that a user is allowed to keep a password; can be set to -1 to remove that limit.
-W *num*	Specifies when a user is warned to change her password, in number of days before the password expiration date.

CERTIFICATION OBJECTIVE 8.02

Administrative Control

It's important for administrators to execute most actions as regular users because the root administrative user has full privileges on a system. Limits on regular users can help protect Linux systems from accidents. Regular users who have the root administrative password can temporarily take root privileges with the **su** command. The **su** command can do more with other accounts. In contrast, the **sg** command is associated with privileges on special groups.

Although the **su** command is adequate for small networks, no administrator should work alone. With the help of the **sudo** command configured in the /etc/sudoers file, it's possible to set up dedicated administrators with partial or complete root administrative privileges, or to execute a command as another user.

The Ability to Log In as root

It's possible to keep users from logging in directly as the root administrative user. Local access is regulated in the /etc/securetty file. By default, it contains access directives for 11 virtual consoles. Although only six virtual consoles are enabled in the /etc/systemd/logind.conf file discussed in Chapter 5, it is possible to configure 12 (based on the number of function keys on a keyboard).

The virtual consoles listed in /etc/securetty determine the consoles where the root administrative user is allowed to log in. If the directives in this file were commented out, administrators would not be able to log in directly to the root account. They'd have to log in to a regular account and use either the **su** or **sudo** command for administration.

Although it's still possible to log in remotely as the root administrative user with the **ssh** command, that ability can also be regulated. The configuration of the SSH server in this manner is an RHCE skill described in Chapter 11.

EXERCISE 8-3

Limit root Logins

In this exercise, you'll examine the effect of eliminating the consoles in the /etc/securetty file. But first, you'll confirm that the root administrative user can log in to the standard consoles on virtual terminals 1 through 6. This exercise assumes that a regular account is available on the local system.

1. Move to the second virtual console. Press CTRL-ALT-F2; alternatively, in a KVM VM, click Send Key | CTRL-ALT-F2. At the *login:* prompt that appears, log in as the root user.
2. Repeat the process with the first, third, fourth, fifth, and sixth virtual consoles. Unless /etc/securetty has already been changed, you should be able to log in as the root user in all of these consoles.
3. Back up the current /etc/securetty file.
4. Open the /etc/securetty file in a text editor. Comment out all of the directives and then save the file.
5. Log out of the console. Try logging back in as the root administrative user. What happens? Repeat the process in other virtual consoles. What happens?
6. Log in to a console as a regular user. Does it work? Run the **su -** command to assume root privileges. Restore the original /etc/securetty file.
7. If a regular user account doesn't exist on the system, you'll have to reboot the system in the rescue target, as discussed in Chapter 5. You'll then be able to restore the /etc/securetty file from the prompt that appears.

The Ability to Log In

Beyond the /etc/securetty file is /etc/security/access.conf, which regulates access by all users. While the default version of this file is completely commented out, the comments provide useful examples. The first example would disallow (with the -) logins to the first virtual console (tty1) to *all* users but root:

```
-:ALL EXCEPT root:tty1
```

Jump ahead in the file. The following line is a slightly more complex example that would disallow access to all users, except users who are members of the wheel group, along with the shutdown and sync users, on *local* (non-networked) logins:

```
-:ALL EXCEPT (wheel) shutdown sync:LOCAL
```

Scroll down further in the file. The following lines (with the +) allow the root user to access the system from three specific remote IP addresses, along with the localhost address:

```
+ : root : 192.168.200.1 192.168.200.4 192.168.200.9
+ : root : 127.0.0.1
```

If you're protecting a system from outside networks, this type of limitation on direct root administrative access makes sense. As long as the **su** or **sudo** command allows it, users who log in remotely as regular users can still elevate their privileges accordingly.

Be aware, the directives in this file are considered in order. So if directives that allow access (with the +) come first, then the following directive denies access to all other users from all other local and remote logins:

```
- : ALL : ALL
```

The Proper Use of the su Command

In some cases, such as Red Hat exams, it's appropriate to log in as the root administrative user. But in real production systems, it's best to log in as a regular user. As a regular user, you can temporarily open a shell with root administrative privileges with the **su** command. Normally, that command prompts for the password of the root administrative user. After you've completed administrative tasks, it's best to log out of the root administrative account; the **exit** command would return to the regular account of that user.

The **su -** command is slightly different because it opens a login shell for the root administrative account. If the password is accepted, it navigates to the root user's home directory, /root, and sets an environment as if the root user had logged in directly.

If you have the password of a second user, you can use the **su -** ***username*** command to log in directly to that account. For example, if you wanted to log in to user dickens' account, you'd run the **su - dickens** command. When you enter her password successfully, the command takes you to the /home/dickens directory.

Finally, the **su -c** command can be used to assume administrative privileges for one command. For example, the following command can be used to modify the first virtual drive on a system (assuming the root administrative password is successfully entered in response to the prompt):

```
$ su -c '/sbin/fdisk /dev/vda'
```

Limit Access to su

As discussed earlier, the ability to log in directly as the root administrative user can be regulated. Further limitations on administrative access are possible. For example, you can limit the users who are allowed to run the **su** command. This requires two basic steps.

First, you'll need to list the users who should have access to the **su** command. Make them a part of the wheel group. By default, here is what this line in /etc/group looks like:

```
wheel:x:10:
```

You can add selected users to the end of this line directly with the **usermod -G wheel *username*** command or with the User Manager.

Second, this requires a change to the configuration of Pluggable Authentication Modules (PAM). While PAM, as described in Chapter 10, is an RHCE objective, there's a commented directive available in the /etc/pam.d/su file ready for this purpose:

```
# auth   required pam_wheel.so use_uid
```

If this line is activated, only users who are members of the wheel group are allowed to use the **su** command.

The Proper Use of the sg Command

With the **sg** command, you can execute another command with the rights associated with a special group. This assumes you're a member of the group or you've set up a group password for the project group with the **gpasswd project** command. Then the command **sg project -c *command*** allows you access to files and directories owned by the group named project. For example, if the /home/secret directory is owned by the project group, the following command copies the important.doc file to the noted directory:

```
$ sg project -c 'cp important.doc /home/project'
```

Custom Administrators with the sudo Command

You can limit access to the **sudo** command. Regular users who are authorized in /etc/sudoers can access administrative commands with their own password. You don't need to give out the administrative password to everyone who thinks they know as much as a Red Hat–certified professional.

To access /etc/sudoers with the editor specified in the /etc/environment file, run the **visudo** command. The **visudo** command locks the /etc/sudoers file against simultaneous edits and checks the syntax of the file before exiting. The following directive is active in the default version of the file. This allows the root user full access to administrative commands:

```
root     ALL=(ALL) ALL
```

Other users can be given administrative access. For example, if you want to allow user boris full administrative access, add the following directive to /etc/sudoers:

```
boris     ALL=(ALL) ALL
```

In this case, all boris needs to do to run an administrative command is to preface it with the **sudo** command. For example, if boris runs the following command, he's prompted for his own regular user password before the noted service is started:

```
$ sudo systemctl start vsftpd
Password:
```

Alternatively, you can allow special users administrative access without a password. As suggested by the comments, the following directive in /etc/sudoers would allow all users who are members of the wheel group to run administrative commands without a password:

```
%wheel   ALL=(ALL)  NOPASSWD: ALL
```

But you don't have to allow full administrative access. For example, if you want to allow users who are members of the %users group to shut down the local system, activate the following directive:

```
%users  localhost=/sbin/shutdown -h now
```

In many Linux configuration files, the % sign in front of a directive specifies a group. Even though the users group has a GID of 100, it's acceptable to make regular users members of that group. For example, another directive shown in comments specifies a group of commands that can be run by users who are members of the %sys group:

```
%sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, ↵
PROCESSES, LOCATE, DRIVERS
```

Each of the directives can be associated with a set of commands. For example, users in the sys group, who are allowed to run PROCESSES directives, can run the commands associated with the following configuration line:

```
Cmnd_Alias PROCESSES = /bin/nice, /bin/kill, /usr/bin/kill, ↵
/usr/bin/killall
```

In a similar fashion, you could set up an admin group of users who are allowed to run these commands with the following directive:

```
%admin ALL = PROCESSES
```

This assumes that groups such as admin exist in the /etc/group and /etc/gshadow files.

Other Administrative Users

Various services may be configured with their own groups of administrative users. For example, examine the following directive from the /etc/cups/cups-files.conf file:

```
SystemGroup sys root
```

Members of groups listed with the **SystemGroup** get administrative privileges on the RHEL 7 print server.

on the job

CUPS is no longer an acronym, to avoid concerns with the word "UNIX" as a trademark. CUPS is still the name of the default Linux print server, however.

CERTIFICATION OBJECTIVE 8.03

User and Shell Configuration

Each user on any Red Hat Enterprise Linux system has an *environment* when logged on to the system. The environment defines directories where Linux looks for programs to run, the look of the login prompt, the terminal type, and more. This section explains how you can configure the default environment for local users. All system-wide shell configuration files are kept in the /etc directory. These files are bashrc, profile, and the scripts in the /etc/profile.d directory. These files and scripts are supplemented by hidden files in each user's home directory, as just described. Let's take a look at these files.

Home Directories and /etc/skel

When a new user is created with standard commands such as **useradd** or utilities such as the User Manager, a default set of configuration files is copied to the user's home directory from the /etc/skel directory.

Home Directory

The home directory is where a user starts when logging in to a RHEL system. The home directory for most users is /home/*username*, where *username* is the user's login name. Every user should normally have write permission in his own home directory, so each user is free to read and write his own files.

/etc/skel

The /etc/skel directory contains default environment files for new accounts. The **useradd** command and the Red Hat User Manager copies these files to the home directory for new users. The contents of /etc/skel may vary. While the standard files in this directory are hidden, administrators are free to add more files for new users. Standard files from one copy of /etc/skel are described in Table 8-9.

exam watch

Linux includes many hidden files that start with a dot (.). To list these files, run the ls -a **command. For example, if you want to list all the files in the /etc/skel directory, run the** ls -a /etc/skel **command.**

TABLE 8-9 Standard Files in the /etc/skel Directory

File	Purpose
.bashrc	This basic bash configuration file may include a reference to the general /etc/bashrc configuration file. It can include commands to run when the bash shell is started. One example is an alias such as **rm='rm -i'**.
.bash_logout	This file is executed when you exit a bash shell, and it can include commands appropriate for this purpose, such as commands for clearing a screen.
.bash_profile	This file is sourced only when invocating a bash login shell, and it configures the bash startup environment. This is the appropriate place to add environment variables or modify the directories in your user account PATH.
.kde/	Specifies settings for the K Desktop Environment. It is not added to /etc/skel and not copied to user home directories if KDE is not installed.
.mozilla/	Includes options associated with the Firefox web browser, developed by the Mozilla project.

If you've installed more than a standard set of software packages on RHEL, additional configuration files and subdirectories may appear in the /etc/skel directory. For example, the installation of certain packages may include configuration files associated with emacs and the Z shell (zsh) in this directory.

As the system administrator, you can edit these files or place custom files in /etc/skel. When new users are created, these files are propagated to the new users' home directories.

/etc/bashrc

The /etc/bashrc file is used for aliases and functions on a system-wide basis. Open this file in the text editor of your choice. Read each line in this file. Even if you don't understand the programming commands, you can see that this file sets the following bash shell parameters for each user:

- It assigns a value of **umask**, which creates the default permissions for newly created files. It supports one set of permissions for root and system users (with user IDs below 200) and another for regular users. (Officially, RHEL reserves all user IDs above 1000 for regular users; however, that is not reflected in /etc/bashrc.)
- It assigns and defines a prompt, which is what you see just before the cursor at the command prompt.
- It includes settings from *.sh files in the /etc/profile.d/ directory.

The settings here are supplemented by the .bashrc file in each user's home directory and, for login shells, by the /etc/profile, .bash_profile, and .bash_logout files.

/etc/profile and /etc/profile.d

The /etc/profile file is used for system-wide environments and startup files, and is sourced when bash is invoked as a login shell.

The first part of the file sets the PATH for searching for commands. Additional directories are added to the PATH with the **pathmunge** command. (Unless you use the Korn shell, ignore the "ksh workaround" stanza.) Then it exports the PATH, USER, LOGNAME, MAIL, HOSTNAME, HISTSIZE, and HISTCONTROL variables and finally sets the umask and runs the scripts in the /etc/profile.d directory. You can check the current value of any of these variables with the **echo $*variable*** command.

/etc/profile.d

The /etc/profile.d directory is designed to contain scripts to be executed in a login or interactive shell (that is, not in a script or a command that is run as **bash -c *command***). If you performed a "Server with GUI" installation, the following is a partial listing of the files; those with .sh extensions apply to the default bash shell:

```
256term.csh                  colorls.csh  PackageKit.sh
256term.sh                   colorls.sh   vim.csh
abrt-cosole-notifiction.sh lang.csh     vim.sh
bash_completion.sh           lang.sh      vte.sh
colorgrep.csh                less.csh     which2.csh
colorgrep.sh                 less.sh      which2.sh
```

In most cases, there are two versions of a script, customized for different shell environments. Look at the files in the /etc/profile.d script directory. You can see that any script in this directory that ends with .sh is included as part of the configuration with /etc/profile. Scripts with other extensions, such as .csh, relate to the C shell.

EXERCISE 8-4

Another Way to Secure a System

One more way to help secure a system is to change the default permissions for new files and directories. In this exercise, you'll reconfigure a system to remove access permissions for default files from other users or groups.

1. Back up the current version of the /etc/bashrc and /etc/profile files.
2. Open the /etc/bashrc file in a text editor. Two lines in the file set the umask. One of the two lines is selected, depending on the **if** statement above them. See if you can determine which value of umask is assigned to an average (non-root) user.
3. The **if** statement tests to see whether the username and group name are the same, and that the UID is greater than 199. In other words, the umask value of 002 is given to regular users. A umask value of 022 is given to system users.
4. Change the first **umask** statement to exclude all permissions for groups and others. In other words, replace the umask of 002 with a umask of 077.
5. Save and exit the file.
6. Repeat steps 2 to 5 for /etc/profile.
7. Log in as a regular, non-privileged user. Use the **touch** command to make a new empty file. Use **ls -l** to verify the permissions on that file.

8. Log in as root. Again, use the **touch** command to make a new empty file and use **ls -l** to verify the permissions on that new file.

   You have just changed the default umask for all regular users. While this is an excellent option for security, it would affect the steps used in other chapters. Therefore, the final step is important.
9. Restore the original versions of /etc/bashrc and /etc/profile from the backup created in Step 1.

Shell Configuration Files in User Home Directories

As described earlier, each user gets a copy of all files from the /etc/skel directory, typically when the account is created. Most of them are hidden, revealed only with commands such as **ls -a**. As users start working with their accounts, more configuration files may be added to their home directories. Some users may work primarily with the default bash shell, whereas others will have additional configuration files related to their GUI desktop environments, such as GNOME.

The default Linux shell is bash, and until recently it was specifically included as the only shell described in associated Red Hat exam objectives. Although bash is no longer specifically included in the objectives, it is the default for RHEL 7.

Login, Logout, and User Switching

While this may seem like a "no-brainer" topic for Linux users with even a couple of days of experience, one of the RHCSA topics is "Log in and switch users in multi-user targets." It includes concepts from different chapters. As discussed in Chapter 5, the multi-user targets are multi-user.target and graphical.target. Virtual terminals are available at all of these targets. For the first RHEL 7 release, a text login prompt appears as follows:

```
Red Hat Enterprise Linux Server 7.0 (Maipo)
Kernel 3.10.0-123.el7.x86_64 on an x86_64

server1 login:
```

The hostname, as well as the versions of RHEL 7 and the kernel, will vary. But that's irrelevant to actual logins; all you need to do is type in a username, press ENTER, and type in a password when prompted.

Logouts from the command line are even simpler; the **exit**, **logout**, and CTRL-D commands all perform logouts from the command line. Of course, once you've logged out from a system, the login prompt just shown will appear.

As discussed earlier in this chapter, there's a different way to switch user accounts. For example, to switch from the current account to user donna's account, run the following command:

```
$ su - donna
```

The same **exit**, **logout**, and CTRL-D commands can be used to exit from user donna's account.

Of course, users can log in to and log out of the GUI. Although the steps vary a bit by desktop environment, they are as simple as the steps required to log in to and log out of any other operating system.

CERTIFICATION OBJECTIVE 8.04

Users and Network Authentication

By default, access to a Linux computer requires a valid username and password. One problem with a large network of Linux systems is that without some central database, each user would require an account on every Linux computer.

exam watch

In the RHCSA exam objectives for RHEL 6, the only network authentication requirement was to be able to connect a client to an LDAP server. The corresponding objective for RHEL 7 is more generic and requires you to "configure a system to use an existing authentication service for user and group information." This may not include just LDAP, but also other services, such as Kerberos, Active Directory, and IPA. The authconfig **tool supports all of them and allows you to configure a client in a few easy steps.**

Several services are available that can be used as a central authentication database. One legacy option for Linux systems is the Network Information Service (NIS). In contrast, the Lightweight Directory Access Protocol (LDAP) offers more security and is now a de facto standard. Other services are available, such as Winbind, which can be configured to support access by Linux systems and users to networks governed by Microsoft Active Directory. Another option is IPA (and its free version FreeIPA), which is an Identity, Policy, and Auditing server that includes a certification authority, along with LDAP and Kerberos

TABLE 8-10 Common Network Authentication Services

Service	User and Group Information	Authentication
Local files	Retrieved from /etc/passwd and /etc/group	Hashed passwords in /etc/shadow
Network Information System (NIS) Server	Centralized /etc/passwd and /etc/group	Centralized /etc/shadow
Network Information System (NIS) Server with MIT Kerberos KDC	Centralized /etc/passwd and /etc/group	Kerberos
OpenLDAP, 389 Directory Server	LDAP/LDAPS protocol	LDAP/LDAPS protocol
OpenLDAP or 389 Directory Server with MIT Kerberos KDC	LDAP/LDAPS protocol	Kerberos
IPA, FreeIPA	LDAP/LDAPS against a 389 Directory Server	Kerberos
Microsoft Active Directory	LDAP/LDAPS protocol	Kerberos

services. In any of these cases, one database of passwords and usernames exists for a network.

Table 8-10 illustrates some of the common options available for centralized authentication as well as the protocols and resources that each solution relies on to retrieve user information and perform authentication.

As you can see from Table 8-10, different solutions can be used for user account information and authentication. As an example, you can use an LDAP server as a database for user and group information, while authentication can be provided by a Kerberos Key Distribution Center (KDC).

The focus of the next section is LDAP as a client. You'll configure a RHEL 7 system as an LDAP client, set up authentication in the name service switch file, and repeat the process with Red Hat network authentication tools. First, we'll show you how you can configure LDAP clients using the command-line interface and then use the Red Hat Authentication Configuration tool to repeat the process. This way, you'll know two methods for configuring LDAP clients.

In contrast to NIS, LDAP services can be configured on a variety of platforms. Of course, LDAP servers can be configured on RHEL 7, but they are not part of the current RHCSA or RHCE exam objectives. LDAP is also used by IPA and Microsoft-based Active Directory (AD) services.

on the job

LDAP directory services and authentication was one focus of the retired Red Hat's RH423 course, whereas the new RH413 course covers identity management with IPA. If you want to set up a centralized LDAP authentication server, explore the 389 Directory Server at http://directory.fedoraproject.org.

LDAP Client Configuration

To configure a RHEL computer as an LDAP client, you'll need the openldap-clients, openldap, and nss-pam-ldapd RPM packages. The openldap-clients and nss-pam-ldapd RPMs are both optional parts of the Directory Client package group. The openldap package should be installed by default on RHEL 7 systems that have the "Server with GUI" environment group selected at installation, as suggested in an earlier lab in this book.

To configure an LDAP client, you'll need to configure various LDAP configuration files—namely, /etc/nslcd.conf and /etc/openldap/ldap.conf. While the files can seem complex, you don't have to reconfigure a lot just to set up an LDAP client.

/etc/nslcd.conf

The default version of the /etc/nslcd.conf file includes a number of different commands and comments. Standard changes required to set up a basic LDAP client are based on several directives shown in Table 8-11. Encryption-related directives in this file may be

TABLE 8-11 Client Configuration Parameters in /etc/nslcd.conf

Directive	Description
uri	Configures the URI for the LDAP server, in the format ldap://*hostname*. The URI scheme ldap:// specifies the use of the LDAP protocol in clear text (on TCP port 389), whereas ldaps:// is for LDAP over SSL (on TCP port 636).
base dc=example,dc=com	Sets the default **base** distinguished name to be used for LDAP searches to retrieve user and group objects (in this case, dc=example,dc=com).
ssl start_tls	Required if StartTLS is used to negotiate an encrypted communication over TCP port 389. As an alternative, encrypted communications can also be provided by turning off StartTLS (**ssl off**) and using LDAP over SSL via the ldaps:// URI scheme.
tls_cacertdir /etc/openldap/cacerts	Specifies the directory where the certificate of the Certification Authority (CA) is stored. This is required when using SSL or TLS for encryption.
nss_initgroups_ignoreusers root	Prevents group lookups for the specified users in the LDAP server.

associated with both the Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS).

The nslcd.conf file applies Pluggable Authentication Modules to LDAP authentication. It is almost identical to the /etc/pam_ldap.conf file from RHEL 6; the differences do not affect the successful configuration of an LDAP client.

Related directives are included at the end of the file; they may include the following. First, the Uniform Resource Identifier, as specified by the **uri**, should redirect the client to the actual IP address of the LDAP server:

```
uri ldap://127.0.0.1/
```

If you want to enable secure communication via LDAP over SSL, you may change **ldap** to **ldaps**; when you configure the LDAP server, those protocols default to TCP/IP ports 389 and 636, respectively. An LDAP server won't work if a firewall blocks these ports. If LDAP over SSL is used, you must either specify the URI using the ldaps:// scheme or change the following to **ssl yes**:

```
ssl no
```

An alternative method to get encrypted communications to an LDAP server is to use StartTLS, which negotiates secure communications over TCP port 389. In this case, you would set **ssl start_tls** and type the URI using the ldap:// scheme.

Of course, to enable secure connections, LDAP needs access to appropriate certificates. While TLS is the successor to SSL, it's commonly used in concert with SSL directives. The following directive specifies the directory with those certificates:

```
tls_cacertdir /etc/openldap/cacerts
```

Finally, the nslcd service must be started and enabled at boot:

```
systemctl enable nslcd
systemctl start nscld
```

/etc/openldap/ldap.conf

You'll need to specify the **URI**, **BASE**, and **TLS_CACERTDIR** variables in this file, just as was done in the /etc/nslcd.conf configuration file. Given the parameters in the preceding section, you may even see a fourth directive in that file:

```
URI ldap://127.0.0.1
SASL_NOCANON    on
BASE dc=example,dc=com
TLS_CACERTDIR /etc/openldap/cacerts
```

If the LDAP server is not on the local system and the base distinguished name is not dc=example,dc=com, substitute accordingly. Individual users can supersede this file in a hidden .ldaprc file in their home directories.

The Name Service Switch File

The Name Service Switch (NSS) file, /etc/nsswitch.conf, governs how a computer searches for key files such as password databases. It can be configured to look through LDAP and other server databases. For example, when a client looks for a computer hostname, it might start with the following entry from /etc/nsswitch.conf:

```
hosts: files ldap dns
```

This line tells your computer to search through name databases in the following order:

1. Start with the database of hostnames and IP addresses in the local file /etc/hosts.
2. Search for the hostname by querying an LDAP server.
3. If none of these databases includes the desired hostname, refer to the DNS server.

You can configure the /etc/nsswitch.conf configuration file to look at an LDAP server for the desired databases. For example, to set up a centralized username and password database for your network, you'll need to configure at least the following commands in /etc/nsswitch.conf:

```
passwd:   files ldap
shadow:   files ldap
group:    files ldap
```

Other authentication databases can be configured; NIS is associated with the **nis** directive; Microsoft authentication can be configured either via LDAP-based AD services or by joining the Linux host to the AD domain with **winbind**. Another important client authentication service is **sssd**, which is the topic of the next section.

The System Security Service Daemon

The System Security Services Daemon (SSSD) provides caching and offline authentication services to allow users to authenticate even when a remote LDAP server is unavailable. SSSD can be used as a replacement for the nss-pam-ldapd daemon. It comes with several related RPM packages, such as sssd-ad, which SSSD can use to fetch identity data from an Active Directory server. You can install these packages, with dependencies, by installing the meta sssd RPM package:

```
yum -y install sssd
```

In a similar way to nss-pam-ldapd, SSSD provides an interface to NSS and PAM. However, SSSD is much more powerful and can also authenticate users with Kerberos, Active Directory, and IPA.

You can find the SSSD configuration file in /etc/sssd/sssd.conf. If the file is not present in your system, **authconfig** (the Red Hat Authentication Configuration tool, covered in the next section) can generate the file for you. A sample configuration for an LDAP client is shown here:

```
id_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://127.0.0.1
ldap_id_use_start_tls = True
ldap_tls_cacertdir = /etc/openldap/cacerts
[sssd]
services = nss, pam
config_file_version = 1
domains = default
```

The first three lines tell SSSD to use LDAP for user information, authentication, and change-password operations. Then, an LDAP URI is specified, similar to the **uri** directive in /etc/nslcd.conf. The next two configuration options enable the use of TLS for encryption and the directory where the CA certificate is stored. Then, SSSD is instructed to work along with NSS and PAM. Finally, at least a default SSSD domain must be configured. A domain name is used to identify different user database information in such cases where more than an authentication method is available on the network.

When SSSD is used to retrieve remote user and authentication information, the entries in /etc/nsswitch.conf should look similar to the lines shown here:

```
passwd:   files sss
shadow:   files sss
group:    files sss
```

Whereas the **ldap** directive in /etc/nsswitch.conf tells the system to use the **nslcd** daemon to look up user information, the **sss** keyword uses SSSD instead. Of course, the SSSD daemon must be started and enabled at boot:

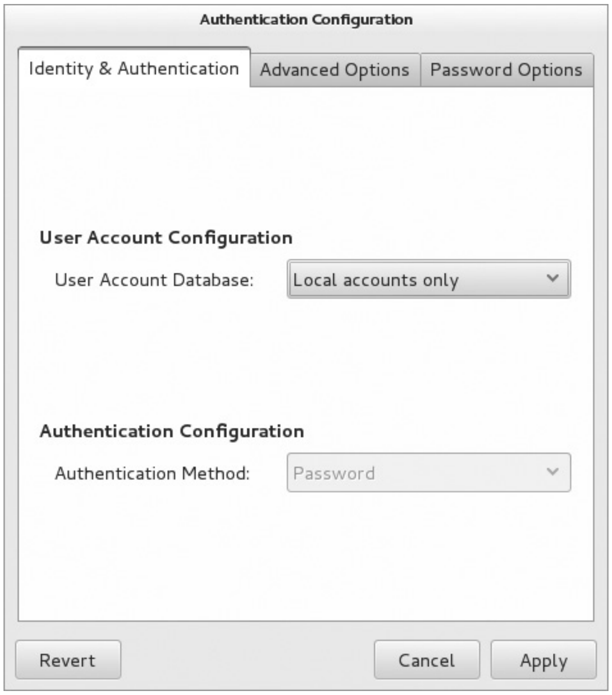
```
# systemctl enable sssd
# systemctl start sssd
```

Red Hat Network Authentication Tools

As you have seen in the previous sections, the configuration of an LDAP client requires you to edit several files. As such, the configuration process can be very error prone if you are not very familiar with all the configuration options. It is certainly easier to configure a client

FIGURE 8-4

Authentication Configuration options



User Account Configuration

User Account Database: Local accounts only

Authentication Configuration

Authentication Method: Password

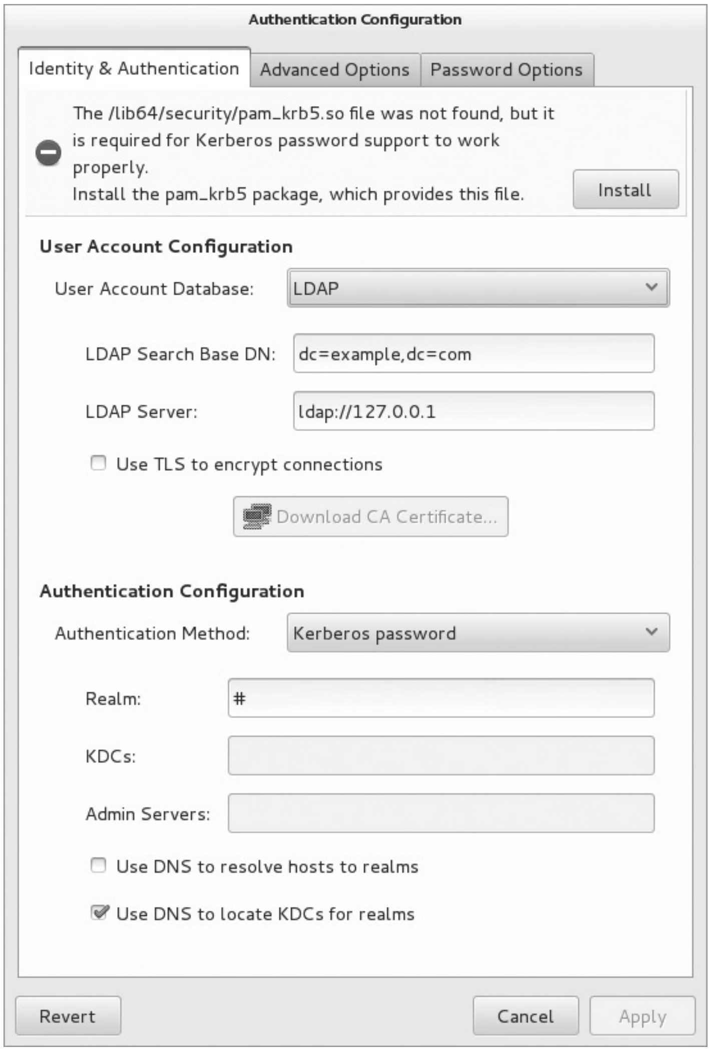
with the Red Hat Authentication Configuration tool. In RHEL 7, you can open it in the GUI with the **system-config-authentication** command or with **authconfig-gtk**. There is also a console version that can be started with the **authconfig-tui** command or the CLI tool **authconfig**. Both the GUI and TUI tools are provided by the authconfig-gtk RPM package. The GUI version is shown in Figure 8-4.

LDAP Client

The Authentication Configuration tool has changed. By default, it's set to look at only the local authentication database, but if you click the drop-down text box, it presents five other options. LDAP is one of the most common protocols for authentication services and user information, but you should also become familiar with the settings of the other options. When LDAP is selected, the window changes, as shown in Figure 8-5. It defaults to the Kerberos Password Authentication Method.

FIGURE 8-5

LDAP Authentication Configuration options


Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

The /lib64/security/pam_krb5.so file was not found, but it is required for Kerberos password support to work properly.
Install the pam_krb5 package, which provides this file.

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: dc=example,dc=com

LDAP Server: ldap://127.0.0.1

☐ Use TLS to encrypt connections

Authentication Configuration

Authentication Method: Kerberos password

Realm: #

KDCs:

Admin Servers:

☐ Use DNS to resolve hosts to realms

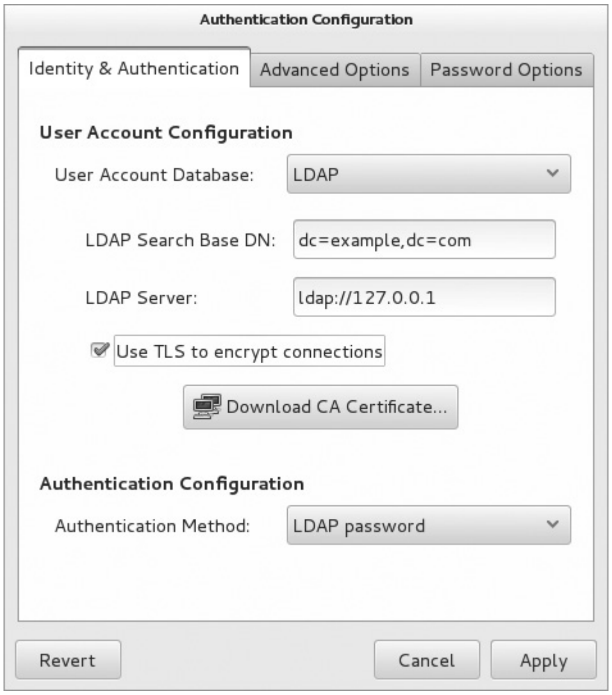
☑ Use DNS to locate KDCs for realms

Note the warning in Figure 8-5. This tells us that in order to use Kerberos as a password authentication method, we need to install the pam_krb5 LDAP package. If a Kerberos server is not available for authentication, click the Authentication Method text box and select LDAP Password. The window changes again and should provide the following warning:

```
You must provide ldaps:// server address or use TLS for LDAP authentication.
```

FIGURE 8-6

LDAP authentication with TLS encryption


Authentication Configuration

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: dc=example,dc=com

LDAP Server: ldap://127.0.0.1

✓ Use TLS to encrypt connections

Authentication Configuration

Authentication Method: LDAP password

If you use LDAPS or StartTLS to encrypt traffic, the warning will disappear. In that case, what you see should resemble Figure 8-6.

The remaining options may vary:

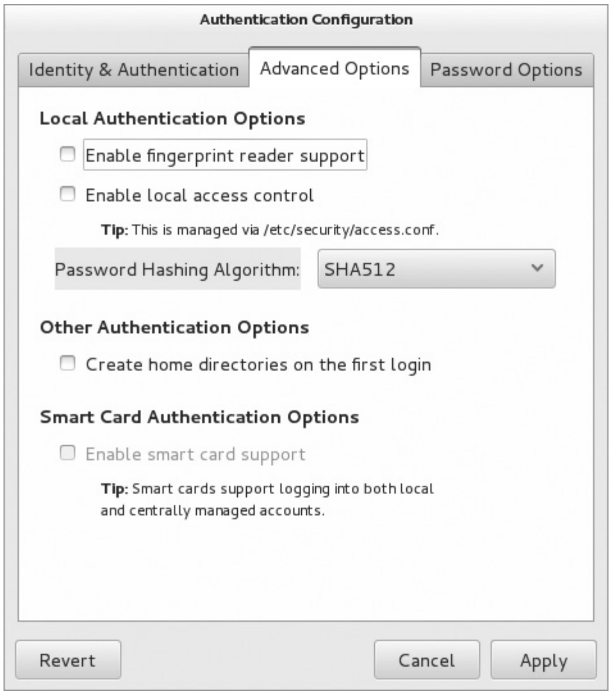
1. The LDAP Search Base DN text box usually includes the domain name for the LDAP server and one or more Organizational Units (ou). For example, if the local system domain is example.com and the users are located under ou=People, the text box may contain the following:

   ```
   ou=People,dc=example,dc=com
   ```

2. The LDAP server text box should include the URI of that server. If your LDAP server is on the local computer, you can use the 127.0.0.1 IP address. But that's unlikely in production situations, which means that it is also unlikely during an exam. For standard LDAP communications, preface the URI with ldap://. For SSL-based LDAP communications, preface the URI with ldaps://. Alternatively, if you use StartTLS, preface the URI with ldap:// and select the Use TLS to Encrypt Connections check box.
3. If you configure secure LDAP, you need to include a Certification Authority (CA) certificate. Click Download CA Certificate. It opens a window where you can specify the URL with the CA certificate.

FIGURE 8-7

Advanced Options tab


Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

Local Authentication Options

☐ Enable fingerprint reader support

☐ Enable local access control

Tip: This is managed via /etc/security/access.conf.

Password Hashing Algorithm: SHA512

Other Authentication Options

☐ Create home directories on the first login

Smart Card Authentication Options

☐ Enable smart card support

Tip: Smart cards support logging into both local and centrally managed accounts.

4. Now select the Advanced Options tab, shown in Figure 8-7. It's not related to the configuration of an LDAP client. In some configurations, you may want to select Create Home Directories on the First Login. That option enables the pam_mkhomedir PAM module to automatically create a user's home directory at first logon, if it does not yet exist.

Once you've made desired changes, click OK; it may take a few seconds for the Authentication Configuration tool to write the changes to the noted configuration files.

IPA Client

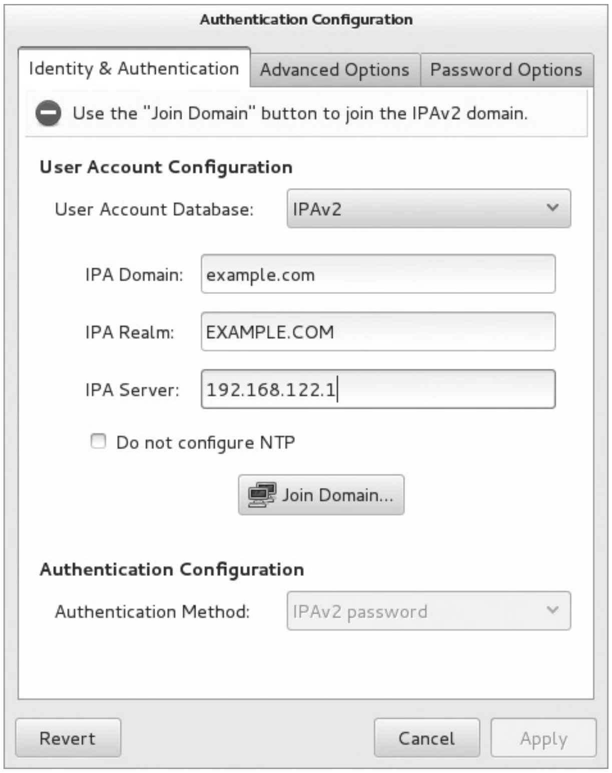
IPA (and its free version FreeIPA) is an identity management suite that includes an LDAP 389 Directory Server, a MIT Kerberos KDC, a Dogtag certification authority, an NTP service, and an optional DNS service. While the configuration of an IPA server is outside the scope of this book and of the RHCSA exam, it is relatively easy to set up an IPA client.

First, install the ipa-client RPM package. This will also install needed dependent packages, such as krb5-workstation and sssd:

```
# yum install ipa-client
```

FIGURE 8-8

IPA Authentication Configuration options


Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

Use the "Join Domain" button to join the IPAv2 domain.

User Account Configuration

User Account Database: IPAv2

IPA Domain: example.com

IPA Realm: EXAMPLE.COM

IPA Server: 192.168.122.1

☐ Do not configure NTP

Join Domain...

Authentication Configuration

Authentication Method: IPAv2 password

Revert | Cancel | Apply

Next, launch **authconfig-gtk**, which is shown in Figure 8-8. The configuration options are straightforward:

- The domain is a DNS domain. As an IPA client, this should correspond to the domain for IPA Identity Management services. For example, if the client is server1.example.com, the corresponding domain would be example.com.
- The realm is a Kerberos realm, and usually is specified as a domain in capital letters, such as EXAMPLE.COM.
- The server text box should include the IP address or FQDN of the server.

Once all the settings have been entered, click Join Domain. You will be prompted for the username and password of an IPA server account with privileges to add new clients to the system.

on the job

If you want to configure an IPA server, refer to the FreeIPA project at www.freeipa.org.

CERTIFICATION OBJECTIVE 8.05

Special Groups

In the past, Linux groups of regular users allowed their members to share files. Red Hat has helped change this with the way it assigns unique UID and GID numbers to each user. When regular users are all made members of the same primary group, that also means everyone in that group has access to the home directories of all other group members—and that's often not desirable. Users may not want to share the files in their home directories with others.

On the other hand, RHEL gives each user a unique user ID and group ID in /etc/passwd. This is known as the *user private group* scheme. Users get exclusive access to their own primary groups and don't have to worry about other users reading the files in their home directories.

Standard and Red Hat Groups

In RHEL, each user gets her own special private group by default. As noted earlier, UIDs and GIDs normally start at 1000, are assigned matching numbers, and proceed in ascending order. In addition, you can set up special groups of dedicated users, ideally with higher GIDs. For example, an administrator might configure accgrp for the accounting department, perhaps with a GID of 70000.

Shared Directories

Most people work in groups, and they may want to share files. However, there may be good reasons for people in those groups to keep their information hidden from others. To support such groups, you can set up a shared directory, with access limited to the members of the group.

Assume you want to set up a shared directory, /home/accshared, for a group of accountants. To that end, you can set up a shared directory with the following basic steps:

1. Create the shared directory:

   ```
   # mkdir /home/accshared
   ```

2. Create a group for the accountants. Call it accgrp. Give it a group ID that doesn't interfere with existing group or user IDs. One way to do this is to add a line such as the following to the /etc/group file or with the User Manager. Substitute the desired usernames.

   ```
   accgrp:x:70000:robertc,alanm,victorb,roberta,alano,charliew
   ```

3. Set up appropriate ownership for the new shared directory. The following commands prevent any specific user from taking control of the directory and assign group ownership to accgrp:

   ```
   # chown nobody.accgrp /home/accshared
   # chmod 2770 /home/accshared
   ```

Any user who is a member of the accgrp group can now create files in and copy files to the /home/accshared directory. Any files generated within or copied to that directory will be owned by the accgrp group.

This is made possible by the 2770 permissions assigned to the /home/accshared directory. Let's break that down into its component parts. The first digit (2) is the *set group ID bit,* also known as the *SGID bit.* When an SGID bit is set on a directory, any files created in that directory automatically have their group ownership set to be the same as the group owner of the directory. In addition, group ownership of files copied from other directories is reassigned (in this case, to the group named accgrp). There is a second way to set the SGID bit for the /home/accshared directory:

```
chmod g+s /home/accshared
```

The remaining digits are basic knowledge for any experienced Linux or Unix user. The **770** sets read, write, and execute permissions for the user and group that own the directory. Other users get no permissions to that directory. However, because the user owner of the directory is the non-privileged user named nobody, the group owner of the directory is most important. In this case, members of the accgrp group have read, write, and execute permissions to files created in this directory.

EXERCISE 8-5

Control Group Ownership with the SGID Bit

In this exercise, you will create new files in a directory designed to be shared by a group of users. You'll also see the difference in what happens before and after the SGID bit is set.

1. Add users called test1, test2, and test3. Specify passwords when prompted. Check the /etc/passwd and /etc/group files to verify that each user's private group was created:

   ```
   # useradd test1; echo changeme | passwd --stdin test1
   # useradd test2; echo changeme | passwd --stdin test2
   # useradd test3; echo changeme | passwd --stdin test3
   ```

2. Edit the /etc/group file and add a group called tg1. Make the test1 and test2 accounts members of this group. You could add the following line to /etc/group directly or use the Red Hat User Manager:

   ```
   tg1:x:99999:test1,test2
   ```

   Before you proceed, make sure the group ID assigned to group tg1 (in this case, 99999) is not already in use. Make sure to add the following line to /etc/gshadow. A group password is not required.

   ```
   tg1:!::test1,test2
   ```

3. Create a directory intended for use by the tg1 group:

   ```
   # mkdir  /home/testshared
   ```

4. Change the user and group ownership of the shared directory:

   ```
   # chown  nobody.tg1  /home/testshared
   ```

5. Log in as user test1. Make sure the login navigates to the /home/test1 directory. Run the **umask** command to confirm that files created from this account will have the appropriate permissions. (The output of the **umask** command for regular users such as test1 should be 0002.) If there's a problem with the home directory or the **umask** output, you may have made an error earlier in this chapter with user settings. If so, repeat Steps 1–5 on a different VM.
6. Run the **cd /home/testshared** command. Now try to create a file with the following commands. What happens?

   ```
   $ date >> test.txt
   $ touch abcd
   ```

7. Now as the root user, set group write permissions on the testshared directory:

   ```
   # chmod 770 /home/testshared
   ```

8. Log in again as user test1, navigate back to the /home/testshared directory, and then try to create a file in the new directory. So far, so good.

   ```
   $ cd /home/testshared
   $ date >> test.txt
   $ ls -l test.txt
   ```

9. Remove all permissions for other users on new files in the /home/testshared directory:

   ```
   # chmod o-rwx /home/testshared/*
   ```

10. Now with the following command, check the ownership on the new file. Do you think other users in the tg1 group can access this file? If in doubt, log in as user test2 and see for yourself.

    ```
    $ ls -l
    ```

11. From the root account, set the SGID bit on the directory:

    ```
    # chmod g+s /home/testshared
    ```

    (Yes, if you are efficiency minded, you may know that the **chmod 2770 /home/testshared** command combines the effect of this and the previous **chmod** commands.)
12. Switch back to the test1 account, navigate back to the /home/testshared directory, and create another file. Remove permissions for other users on the newly created file. Check the ownership on the newly created file. Do you think that user test2 can now access this file? (To see for yourself, try it from the test2 account.)

    ```
    $ date >> testb.txt
    $ chmod o-rwx /home/testshared/testb.txt
    $ ls -l
    ```

13. Now log in as the test2 account. Go into the /home/testshared directory. Try accessing the testb.txt file. Create a different file and then use **ls -l** to check permissions and ownership again. (To see that it worked, try accessing this file from the test1 account.)
14. Switch to the test3 account and check whether that user can or cannot create files in this directory and whether that user can or cannot view the files in this directory.

CERTIFICATION SUMMARY

You can manage users and groups with the files of the shadow password suite. These files can be modified directly, with the help of commands such as **useradd** and **groupadd** or the User Manager tool. The way users are configured is based on the /etc/login.defs file. Any variables or system-wide settings are defined in /etc/bashrc or /etc/profile. They can be modified by files in user home directories.

There are several ways to limit the use of administrative privileges. The ability to log in can be regulated in files such as /etc/securetty and /etc/security/access.conf. Access to the **su** command can be limited with the help of PAM. Partial and complete administrative privileges can be configured for the **sudo** command in the /etc/sudoers file.

You can use centralized network account management with the LDAP service. RHEL 7 systems can be configured as an LDAP client with the help of the /etc/nslcd.conf, /etc/openldap/ldap.conf, and /etc/nsswitch.conf files.

By default, Red Hat Enterprise Linux assigns unique user and group ID numbers to each new user. This is known as the user private group scheme. This scheme supports the configuration of special groups for a specific set of users. The users in the group can be configured with read and write privileges in a dedicated directory, courtesy of the SGID bit.

✓ TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 8.

User Account Management

❑ After installation, a system may have only a single login account: root. For everyday operation, it's best to create one or more regular accounts.

❑ The shadow password suite is configured in the /etc/passwd, /etc/shadow, /etc/group, and /etc/gshadow files.

❑ Administrators can add user and group accounts by directly editing the files of the shadow password suite, or with commands such as **useradd** and **groupadd**. The way accounts are added are defined by the /etc/login.defs file.

❑ Accounts can be added with the Red Hat User Manager tool. You can also use this tool or related commands such as **chage** and **usermod** to modify other account parameters.

Administrative Control

❑ Logins as the root user can be regulated by the /etc/securetty file.

❑ Logins in general can be regulated by the /etc/security/access.conf file.

❑ Access to the **su** command can be regulated through the /etc/pam.d/su file.

❑ Custom administrative privileges can be configured in the /etc/sudoers file.

User and Shell Configuration

❑ The home directory for new login accounts is populated from the /etc/skel directory.

❑ Each user has an environment when logged on to the system, based on /etc/bashrc, /etc/profile, and the scripts in /etc/profile.d/.

❑ All users have hidden shell configuration files in their home directories.

Users and Network Authentication

❑ LDAP allows you to configure one centrally managed username and password database with other Linux and Unix systems on a LAN.

❑ LDAP clients are configured in /etc/openldap/ldap.conf and either /etc/nslcd.conf (for the nslcd daemon) or /etc/sssd/sssd.conf (for SSSD).

- ❑ Changes are required to /etc/nsswitch.conf to make a system look for a remote authentication database such as LDAP.
- ❑ Red Hat includes **authconfig-gtk** and **authconfig-tui**, two GUI and console tools that can help you configure a system as an LDAP or IPA client.

Special Groups

- ❑ Red Hat's user private group scheme configures users with their own unique user and group ID numbers.
- ❑ With appropriate SGID permissions, you can configure a shared directory for a specific group of users.
- ❑ Setting the SGID bit is easy; use **chown** to set nobody as the user owner and the name of the group as the group owner. Then run the **chmod 2770** command on the shared directory.

SELF TEST

The following questions will help 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. It is okay if you have another way of performing a task. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of these questions.

User Account Management

1. What's the standard minimum user ID number for regular users on Red Hat distributions?

2. What command at a GUI-based text console starts the Red Hat User Manager?

Administrative Control

3. What file regulates the local consoles where the root user can log in?

   ____

4. Which file controls the commands that a user can run with the privileges of root or of another user?

   ____

5. When a regular user uses the **sudo** command to run an administrative command, what password is required?

   ____

User and Shell Configuration

6. If you want to add files to every new user account, what directory should you use?

   ____

7. What are the system-wide configuration files associated with the bash shell?

   ____

Users and Network Authentication

8. If user objects are located in the Organizational Unit named "People," within another organizational Unit named "Global," which is part of the LDAP domain dc=example, dc=org, what is the LDAP Search Base DN?

   ____

9. What's the full path to the file that refers to an LDAP database for authentication?

   ____

Special Groups

10. What command would set the SGID bit on the /home/developer directory?

    ____

11. What command would set up ownership of the developer group on the /home/developer directory?

    ____

12. What command would add user alpha to the developer group? (This question assumes the alpha user and the developer group already exist and that alpha belongs to no group other than his own.)

____

LAB QUESTIONS

Red Hat presents its exams electronically. For that reason, the labs in this chapter are available from the DVD that accompanies the book, in the Chapter8/ subdirectory. They're available in .doc, .html, and .txt formats, to reflect standard options associated with electronic delivery on a live RHEL 7 system. In case you haven't yet set up RHEL 7 on a system, refer to the first lab of Chapter 2 for installation instructions. The answers for these labs follow the Self Test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

User Account Management

1. The minimum user ID number for regular users on Red Hat distributions is 1000.

2. The command in a GUI-based text console that starts the Red Hat User Manager is **authconfig-gtk** or **system-config-users**.

Administrative Control

3. The file that regulates the local consoles where the root user can log in is /etc/securetty.

4. The file that controls which commands a user can run with the privileges of root or of another user is /etc/sudoers.

5. When a regular user uses the **sudo** command to run an administrative command, the regular password of that user is required, unless the NOPASSWD directive was specified in /etc/sudoers.

User and Shell Configuration

6. To automatically add files to every new user account, you should use the /etc/skel directory.
7. The system-wide configuration files associated with the bash shell are /etc/bashrc, /etc/profile, and the scripts in /etc/profile.d/.

Users and Network Authentication

8. The LDAP Search Base DN is ou=People,ou=Global,dc=example,dc=org.
9. The full path to the file that points to an LDAP database for authentication is /etc/nsswitch.conf.

Special Groups

10. The command that would set the SGID bit on the /home/developer directory is **chmod g+s /home/developer**. Numeric options such as **chmod 2770 /home/developer** are not correct, as they go beyond just setting the SGID bit.
11. The command that sets up ownership of the developer group on the /home/developer directory is **chgrp developer /home/developer**.
12. The command that adds user alpha to the developer group is **usermod -aG developer alpha**.

LAB ANSWERS

Lab 1

While there are a number of methods available to create new users and groups, they should all come to the same result.

1. The output to the **ls -l /home** command should include the following output, substituting today's date:

```
drwx------.   4 newguy   newguy    4096 Jan 19 12:13 newguy
drwx------.   4 intern   intern    4096 Jan 19 12:13 intern
```

2. Run the **ls -la /etc/skel** command. The output should include a number of hidden files, owned by the user root and the group root.
3. Run the **ls -la /home/newguy** and **ls -la /home/intern** commands. The output should include the same hidden files as in /etc/skel, but owned by the users associated with each home directory.

4. The end of the /etc/passwd and /etc/shadow files should include entries for both users. If you've set up a password for these users, it should be in encrypted format in the second column of /etc/shadow.
5. The following entry should exist somewhere in the middle of the /etc/group file. It is acceptable if other users are included at the end of the line.

   ```
   users:x:100:newguy
   ```

6. The following line should be near to or at the end of the /etc/group file; the order of the users in the fourth column does not matter.

   ```
   peons:x:123456:newguy,intern
   ```

Lab 2

The simplest way to limit root logins to the sixth virtual console is in the /etc/securetty file. The only active directives in that file should be

```
vc/6
tty6
```

Of course, there are other ways to do just about anything in Linux. To try it out, press CTRL-ALT-F1 and try logging in as the root user. Press CTRL-ALT-F2, and repeat the process through virtual terminal 6.

Lab 3

Use the answer to the first part of Lab 1 as guidance to verify the ownership and permissions of the /home/senioradm directory, along with the files therein. With respect to **sudo** privileges, you should see the following line in the /etc/sudoers file:

```
senioradm    ALL=(ALL)       ALL
```

To test the result, log in as the senioradm user and run an administrative command, prefaced by a **sudo**. For example, try the following command:

```
# sudo firewall-config
```

Unless you've run the **sudo** command in the last few minutes, this action will prompt for a password. Enter the password created for user senioradm. It should open the Firewall Configuration tool.

Lab 4

Use the answer to Lab 1 as guidance to verify the ownership and permissions of the /home/junioradm directory, along with the files therein. With respect to **sudo** privileges, you should see the following line in the /etc/sudoers file:

```
junioradm    ALL=/usr/sbin/fdisk
```

Next, try running the **fdisk** command:

```
$ sudo /usr/sbin/fdisk -l
```

You'll be prompted for a password. Enter the password created for user junioradm. Unless the passwords are identical, the root password would not work. If successful, you should see a list of partitions for connected drives in the output.

Lab 5

Use the answer to Lab 1 as guidance to verify the ownership and permissions of the /home/infouser directory, along with the files therein. If you're successful, that directory will include an info-*/ subdirectory. In addition, the /etc/skel directory should include an info-*/ subdirectory. That subdirectory should have the same files as those shown in the /usr/share/doc/info-* directory. Of course, that works only if you copy the contents of the info-*/ subdirectory from the /usr/share/doc directory to /etc/skel.

Lab 6

This is a straightforward process, using the following basic steps:

1. Create accounts for mike, rick, terri, and maryam if required. You can use the **useradd** command, edit the /etc/passwd file directly, or work through the User Manager.
2. Set up a group for these users. Configure a group ID outside the range of regular users in /etc/group with a line such as this:

   ```
   galley:x:88888:mike,rick,terri,maryam
   ```

3. Create the /home/galley directory. Give it proper ownership and permissions with the following commands:

   ```
   # mkdir /home/galley
   # chown nobody.galley
   # chmod 2770 /home/galley
   ```