Chapter 3. Managing Users and Groups

The control of users and groups is a core element of Red Hat Enterprise Linux system administration. This chapter explains how to add, manage, and delete users and groups in the graphical user interface and on the command line, and covers advanced topics, such as creating group directories.

3.1. Introduction to Users and Groups

While users can be either people (meaning accounts tied to physical users) or accounts which exist for specific applications to use, groups are logical expressions of organization, tying users together for a common purpose. Users within a group share the same permissions to read, write, or execute files owned by that group.

Each user is associated with a unique numerical identification number called a *user ID* (UID). Likewise, each group is associated with a *group ID* (GID). A user who creates a file is also the owner and group owner of that file. The file is assigned separate read, write, and execute permissions for the owner, the group, and everyone else. The file owner can be changed only by **root**, and access permissions can be changed by both the **root** user and file owner.

Additionally, Red Hat Enterprise Linux supports *access control lists* (ACLs) for files and directories which allow permissions for specific users outside of the owner to be set.

3.1.1. User Private Groups

Red Hat Enterprise Linux uses a *user private group* (*UPG*) scheme, which makes UNIX groups easier to manage. A user private group is created whenever a new user is added to the system. It has the same name as the user for which it was created and that user is the only member of the user private group.

User private groups make it safe to set default permissions for a newly created file or directory, allowing both the user and *the group of that user* to make modifications to the file or directory.

The setting which determines what permissions are applied to a newly created file or directory is called a *umask* and is configured in the **/etc/bashrc** file. Traditionally on UNIX-based systems, the **umask** is set to **022**, which allows only the user who created the file or directory to make modifications. Under this scheme, all other users, *including members of the creator's group*, are not allowed to make any modifications. However, under the UPG scheme, this " group protection" is not necessary since every user has their own private group.

A list of all groups is stored in the **/etc/group** configuration file.

3.1.2. Shadow Passwords

In environments with multiple users, it is very important to use *shadow passwords* provided by the *shadow-utils* package to enhance the security of system authentication files. For this reason, the installation program enables shadow passwords by default.

The following is a list of the advantages shadow passwords have over the traditional way of

storing passwords on UNIX-based systems:

- Shadow passwords improve system security by moving encrypted password hashes from the world-readable **/etc/passwd** file to **/etc/shadow**, which is readable only by the **root** user.
- Shadow passwords store information about password aging.
- Shadow passwords allow the **/etc/login.defs** file to enforce security policies.

Most utilities provided by the *shadow-utils* package work properly whether or not shadow passwords are enabled. However, since password aging information is stored exclusively in the **/etc/shadow** file, some utilities and commands do not work without first enabling shadow passwords:

3.2. Managing Users in a Graphical Environment

The **Users** utility allows you to view, modify, add, and delete local users in the graphical user interface.

3.2.1. Using the Users Settings Tool

Press the **Super** key to enter the Activities Overview, type **Users** and then press **Enter**. The **Users** settings tool appears. The **Super** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the Spacebar. Alternatively, you can open the **Users** utility from the **Settings** menu after clicking your user name in the top right corner of the screen.

To make changes to the user accounts, first select the **Unlock** button and authenticate yourself as indicated by the dialog box that appears. Note that unless you have superuser privileges, the application will prompt you to authenticate as **root**. To add and remove users, select the **+** and **-** button respectively. To add a user to the administrative group **wheel**, change the **Account Type** from **Standard** to **Administrator**. To edit a user's language setting, select the language and a drop-down menu appears.

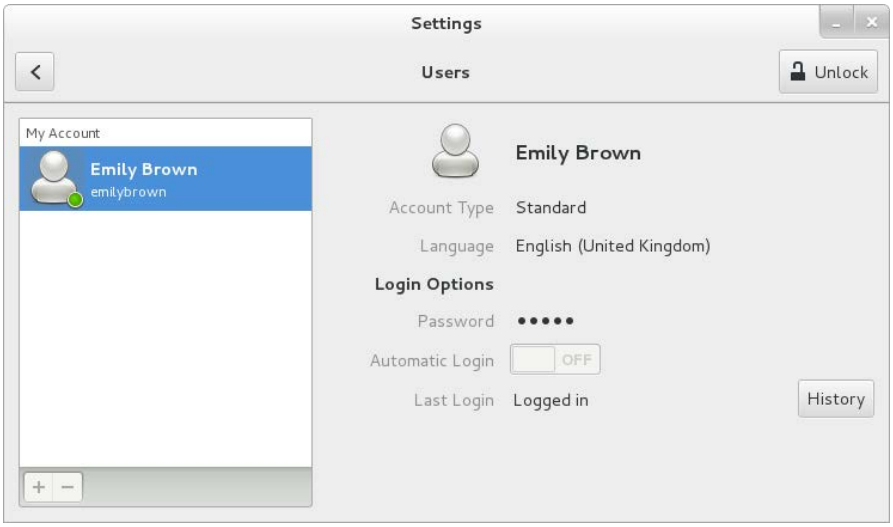

Figure 3.1. The Users Settings Tool

When a new user is created, the account is disabled until a password is set. The **Password** drop-down menu, shown in Figure 3.2, "The Password Menu", contains the options to set a password by the administrator immediately, choose a password by the user at the first login, or create a guest account with no password required to log in. You can also disable or enable an account from this menu.

Figure 3.2. The Password Menu

3.3. Using Command Line Tools

Apart from the **Users** settings tool described, you can use command line tools for managing users and groups that are listed below.

Table 3.1. Command line utilities for managing users and groups

Utilities	Description

id	Displays user and group IDs.
useradd, **usermod**, **userdel**	Standard utilities for adding, modifying, and deleting user accounts.
groupadd, **groupmod**, **groupdel**	Standard utilities for adding, modifying, and deleting groups.
gpasswd	Standard utility for administering the **/etc/group** configuration file.
pwck, **grpck**	Utilities that can be used for verification of the password, group, and associated shadow files.
pwconv, **pwunconv**	Utilities that can be used for the conversion of passwords to shadow passwords, or back from shadow passwords to standard passwords.
grpconv, **grpunconv**	Similar to the previous, these utilities can be used for conversion of shadowed information for group accounts.

3.3.1. Adding a New User

To add a new user to the system, type the following at a shell prompt as **root**:

useradd [*options*] *username*

...where *options* are command-line options as described in Table 3.2, "Common useradd command-line options".

By default, the **useradd** command creates a locked user account. To unlock the account, run the following command as **root** to assign a password:

passwd *username*

Optionally, you can set a password aging policy. See the Password Security section in the *Red Hat Enterprise Linux 7 Security Guide*.

Table 3.2. Common useradd command-line options

Option	Description
-c '*comment*'	*comment* can be replaced with any string. This option is generally used to specify the full name of a user.
-d *home_directory*	Home directory to be used instead of default **/home/*username*/**.
-e *date*	Date for the account to be disabled in the format YYYY-MM-DD.
-f *days*	Number of days after the password expires until the account is disabled. If **0** is specified, the account is disabled immediately after the password expires. If **-1** is specified, the account is not disabled after the password expires.
-g *group_name*	Group name or group number for the user's default (primary) group. The group must exist prior to being specified here.
-G *group_list*	List of additional (supplementary, other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here.
-m	Create the home directory if it does not exist.
-M	Do not create the home directory.
-N	Do not create a user private group for the user.
-p *password*	The password encrypted with **crypt**.
-r	Create a system account with a UID less than 1000 and without a home directory

Explaining the Process

The following steps illustrate what happens if the command **useradd juan** is issued on a system that has shadow passwords enabled:

1. A new line for **juan** is created in **/etc/passwd**:

```
juan:x:1001:1001::/home/juan:/bin/bash
```

The line has the following characteristics:

- It begins with the user name **juan**.
- There is an **x** for the password field indicating that the system is using shadow passwords.
- A UID greater than 999 is created. Under Red Hat Enterprise Linux 7, UIDs below 1000 are reserved for system use and should not be assigned to users.
- A GID greater than 999 is created. Under Red Hat Enterprise Linux 7, GIDs below 1000 are reserved for system use and should not be assigned to users.
- The optional *GECOS* information is left blank. The GECOS field can be used to provide additional information about the user, such as their full name or phone number.
- The home directory for **juan** is set to **/home/juan/**.
- The default shell is set to **/bin/bash**.

2. A new line for **juan** is created in **/etc/shadow**:

```
juan:!!:14798:0:99999:7:::
```

The line has the following characteristics:

- It begins with the username **juan**.
- Two exclamation marks (**!!**) appear in the password field of the **/etc/shadow** file, which locks the account. If an encrypted password is passed using the **-p** flag, it is placed in the **/etc/shadow** file on the new line for the user.
- The password is set to never expire.

3. A new line for a group named **juan** is created in **/etc/group**:

```
juan:x:1001:
```

A group with the same name as a user is called a *user private group*. For more information on user private groups.

The line created in **/etc/group** has the following characteristics:

- It begins with the group name **juan**.
- An **x** appears in the password field indicating that the system is using shadow group passwords.
- The GID matches the one listed for **juan**'s primary group in **/etc/passwd**.

4. A new line for a group named **juan** is created in **/etc/gshadow**:

```
juan:!::
```

The line has the following characteristics:

- It begins with the group name **juan**.
- An exclamation mark (**!**) appears in the password field of the **/etc/gshadow** file, which locks the group.

- All other fields are blank.

5. A directory for user **juan** is created in the **/home** directory:

```
~]# ls -ld /home/juan
drwx------.  4 juan juan 4096 Mar  3 18:23 /home/juan
```

This directory is owned by user **juan** and group **juan**. It has *read*, *write*, and *execute* privileges *only* for the user **juan**. All other permissions are denied.

6. The files within the **/etc/skel/** directory (which contain default user settings) are copied into the new **/home/juan/** directory:

```
~]# ls -la /home/juan
total 28
drwx------.  4 juan juan 4096 Mar  3 18:23 .
drwxr-xr-x.  5 root root 4096 Mar  3 18:23 ..
-rw-r--r--.  1 juan juan   18 Jun 22  2010
.bash_logout
-rw-r--r--.  1 juan juan  176 Jun 22  2010
.bash_profile
-rw-r--r--.  1 juan juan  124 Jun 22  2010 .bashrc
drwxr-xr-x.  4 juan juan 4096 Nov 23 15:09 .mozilla
```

At this point, a locked account called **juan** exists on the system. To activate it, the administrator must next assign a password to the account using the **passwd** command and, optionally, set password aging guidelines .

3.3.2. Adding a New Group

To add a new group to the system, type the following at a shell prompt as **root**: **groupadd** [*options*] *group_name*

...where *options* are command-line options as described.

Table 3.3. Common groupadd command-line options

Option	Description
-f, **--force**	When used with **-g** *gid* and *gid* already exists, **groupadd** will choose another unique *gid* for the group.
-g *gid*	Group ID for the group, which must be unique and greater than 999.
-K, **--key** *key=value*	Override **/etc/login.defs** defaults.
-o, **--non-unique**	Allows creating groups with duplicate GID.
-p, **--password** *password*	Use this encrypted password for the new group.
-r	Create a system group with a GID less than 1000.

3.3.3. Creating Group Directories

System administrators usually like to create a group for each major project and assign people to the group when they need to access that project's files. With this traditional scheme, file management is difficult; when someone creates a file, it is associated with the primary group to which they belong. When a single person works on multiple projects, it becomes difficult to associate the right files with

the right group. However, with the UPG scheme, groups are automatically assigned to files created within a directory with the *setgid* bit set. The setgid bit makes managing group projects that share a common directory very simple because any files a user creates within the directory are owned by the group that owns the directory.

For example, a group of people need to work on files in the **/opt/myproject/** directory. Some people are trusted to modify the contents of this directory, but not everyone.

1. As **root**, create the **/opt/myproject/** directory by typing the following at a shell prompt:

   mkdir /opt/myproject

2. Add the **myproject** group to the system:

   groupadd myproject

3. Associate the contents of the **/opt/myproject/** directory with the **myproject** group:

   chown root:myproject /opt/myproject

4. Allow users in the group to create files within the directory and set the setgid bit:

   chmod 2775 /opt/myproject

   At this point, all members of the **myproject** group can create and edit files in the **/opt/myproject/** directory without the administrator having to change file permissions every time users write new files. To verify that the permissions have been set correctly, run the following command:

   ```
   ~]# ls -ld /opt/myproject
   drwxrwsr-x. 3 root myproject 4096 Mar  3 18:31 /opt/myproject
   ```

5. Add users to the **myproject** group:

   usermod -aG myproject *username*

Gaining Privileges

System administrators, and in some cases users, need to perform certain tasks with administrative access. Accessing the system as the **root** user is potentially dangerous and can lead to widespread damage to the system and data. This chapter covers ways to gain administrative privileges using setuid programs such as **su** and **sudo**. These programs allow specific users to perform tasks which would normally be available only to the **root** user while maintaining a higher level of control and system security.

4.1. The su Command

When a user executes the **su** command, they are prompted for the **root** password and, after authentication, are given a **root** shell prompt.

Once logged in using the **su** command, the user **is** the **root** user and has absolute administrative access to the system. Note that this access is still subject to the restrictions imposed by SELinux, if it is enabled. In addition, once a user has become **root,** it is possible for them to use the **su** command to change to any other user on the system without being prompted for a password.

Because this program is so powerful, administrators within an organization may want to limit who has access to the command.

One of the simplest ways to do this is to add users to the special administrative group called *wheel*. To do this, type the following command as **root**:

```
~]# usermod -a -G wheel username
```

In the previous command, replace *username* with the user name you want to add to the **wheel** group.

You can also use the **Users** settings tool to modify group memberships, as follows. Note that you need administrator privileges to perform this procedure.

1. Press the **Super** key to enter the Activities Overview, type **Users** and then press **Enter**. The **Users** settings tool appears. The **Super** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the **Spacebar**.

2. To enable making changes, click the **Unlock** button, and enter a valid administrator password.

3. Click a user icon in the left column to display the user's properties in the right-hand pane.

4. Change the **Account Type** from **Standard** to **Administrator**. This will add the user to the **wheel** group

After you add the desired users to the **wheel** group, it is advisable to only allow these specific users to use the **su** command. To do this, edit the *Pluggable Authentication Module* (PAM) configuration file for **su**, **/etc/pam.d/su**. Open this file in a text editor and uncomment the following line by removing the # character:

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

This change means that only members of the administrative group **wheel** can switch to another user using the **su** command.

4.2. The sudo Command

The **sudo** command offers another approach to giving users administrative access. When trusted users precede an administrative command with **sudo**, they are prompted for *their own* password. Then, when they have been authenticated and assuming that the command is permitted, the administrative command is executed as if they were the **root** user.

The basic format of the **sudo** command is as follows:

```
sudo command
```

In the above example, *command* would be replaced by a command normally reserved for the **root** user, such as **mount**.

The **sudo** command allows for a high degree of flexibility. For instance, only users listed in the **/etc/sudoers** configuration file are allowed to use the **sudo** command and the command is executed in *the user's* shell, not a **root** shell. This means the **root** shell can be completely disabled as shown in the *Red Hat Enterprise Linux 7 Security Guide*.

Each successful authentication using the **sudo** command is logged to the file **/var/log/messages** and the command issued along with the issuer's user name is logged to the file **/var/log/secure**. If additional logging is required, use the **pam_tty_audit** module to enable TTY auditing for specified users by adding the following line to your **/etc/pam.d/system-auth** file:

```
session required pam_tty_audit.so disable=pattern enable=pattern
```

where *pattern* represents a comma-separated listing of users with an optional use of globs. For example, the following configuration will enable TTY auditing for the **root** user and disable it for all other users:

```
session required pam_tty_audit.so disable=* enable=root
```

Configuring the **pam_tty_audit** PAM module for TTY auditing records only TTY input. This means that, when the audited user logs in, **pam_tty_audit** records the exact keystrokes the user makes into the **/var/log/audit/audit.log** file.

Assignment
1 Mark Questions

1.Define a user in a Linux environment?
2.Define a group in a Linux environment?
3.UID stands for _____
4.GID stands for_____
5.Which command can be used to login as a different user?
6.____ is a special administrate group in a RedHat Linux?
7.What is the use of sudo command in RedHat Linux?

7 Mark Questions

1.What is shadow password? What is its advantages explain?
2.Explain the procedure to create a active user in a Linux system?
3.Explain the procedure to create a group in the system and adding a user in that group?
4.What is a significance of group directories in Linux environment explain?
5.Demonstrate the procedure to create a group directory and set its GID group So that group of directory work on particular directory?
6.Explain the procedure to restrict the access to su command only to few users?
7.What is the use of sudo command in RedHat Linux?