Introduction

Before we proceed, it would be best to cover some basic user administration topics that will be very useful in later chapters. Adding Users

One of the most important activities in administering a Linux box is the addition of users. Here you'll find some simple examples to provide a foundation for future chapters. It is not intended to be comprehensive, but is a good memory refresher. You can use the command man useradd to get the help pages on adding users with the useradd command or the man usermod to become more familiar with modifying users with the usermod command.

Who Is the Super User?

The super user with unrestricted access to all system resources and files in Linux is the user named root. This user has a user ID, of 0 which is universally identified by Linux applications as belonging to a user with supreme privileges. You will need to log in as user root to add new users to your Linux server.

Debian Note: When installing Ubuntu Linux systems, you are prompted to create a primary user that is not `root`. A `root` user is created but no password is set, so you initially cannot log in as this user. The primary user can become the root user using the `sudo su -` command that will be discussed later.

How To Add Users

Adding users takes some planning; read through these steps below before starting:

1) Arrange your list of users into groups by function. In this example there are three groups "parents", "children" and "soho".

```
Parents    Children    Soho

Paul       Alice       Accounts
Jane       Derek       Sales
```

2) Add the Linux groups to your server:

```
[root@bigboy tmp]# groupadd parents
[root@bigboy tmp]# groupadd children
[root@bigboy tmp]# groupadd soho
```

3) Add the Linux users and assign them to their respective groups

```
[root@bigboy tmp]# useradd -g parents paul
[root@bigboy tmp]# useradd -g parents jane
[root@bigboy tmp]# useradd -g children derek
[root@bigboy tmp]# useradd -g children alice
[root@bigboy tmp]# useradd -g soho accounts
[root@bigboy tmp]# useradd -g soho sales
```

If you don't specify the group with the -g, RedHat/Fedora Linux creates a group with the same name as the user you just created; this is also known as the User Private Group Scheme. When each new user first logs in, they are prompted for their new permanent password.

4) Each user's personal directory is placed in the /home directory. The directory name will be the same as their user name.

```
 [root@bigboy tmp]# ll /home
drwxr-xr-x    2 root     root        12288 Jul 24 20:04 lost+found
drwx------    2 accounts soho         1024 Jul 24 20:33 accounts
drwx------    2 alice    children     1024 Jul 24 20:33 alice
drwx------    2 derek    children     1024 Jul 24 20:33 derek
drwx------    2 jane     parents      1024 Jul 24 20:33 jane
drwx------    2 paul     parents      1024 Jul 24 20:33 paul
drwx------    2 sales    soho         1024 Jul 24 20:33 sales
[root@bigboy tmp]#
```

How to Change Passwords

You need to create passwords for each account. This is done with the passwd command. You are prompted once for your old password and twice for the new one.

- User root changing the password for user paul.

```
[root@bigboy root]# passwd paul
Changing password for user paul.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@bigboy root]#
```

- Users might wish to change their passwords at a future date. Here is how unprivileged user paul would change his own password.

```
[paul@bigboy paul]$ passwd
Changing password for paul
Old password: your current password
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password: your new password
Re-enter new password: your new password
Password changed.
[paul@bigboy paul]$
```

How to Delete Users

The userdel command is used to remove the user's record from the /etc/passwd and /etc/shadow used in the login process. The command has a single argument, the username.

```
[root@bigboy tmp]# userdel paul
```

There is also an optional -r switch that additionally removes all the contents of the user's home directory. Use this option with care. The data in a user's directory can often be important even after the person has left your company.

```
[root@bigboy tmp]# userdel -r paul
```

How to Tell the Groups to Which a User Belongs

Use the groups command with the username as the argument.

```
[root@bigboy root]# groups paul
paul : parents
[root@bigboy root]#
```

How to Change the Ownership of a File

You can change the ownership of a file with the chown command. The first argument is the desired username and group ownership for the file separated by a colon (:) followed by the filename. In the next example we change the ownership of the file named text.txt from being owned by user root and group root to being owned by user testuser in the group users:

```
[root@bigboy tmp]# ll test.txt
-rw-r--r--  1 root root 0 Nov 17 22:14 test.txt
[root@bigboy tmp]# chown testuser:users test.txt
[root@bigboy tmp]# ll test.txt
-rw-r--r--  1 testuser users 0 Nov 17 22:14 test.txt
[root@bigboy tmp]#
```

You can also use the chown command with the -r switch for it to doe recursive searches down into directories to change permissions.

Using sudo

If a server needs to be administered by a number of people it is normally not a good idea for them all to use the root account. This is because it becomes difficult to determine exactly who did what, when and where if everyone logs in with the same credentials. The sudo utility was designed to overcome this difficulty.

The sudo utility allows users defined in the /etc/sudoers configuration file to have temporary access to run commands they would not normally be able to due to file permission restrictions. The commands can be run as user "root" or as any other user defined in the /etc/sudoers configuration file.

The privileged command you want to run must first begin with the word sudo followed by the command's regular syntax. When running the command with the sudo prefix, you will be prompted for your regular password before it is executed. You may run other privileged commands using sudo within a five-minute period without being re-prompted for a password. All commands run as sudo are logged in the log file /var/log/messages.

Simple Sudo Examples

Using sudo is relatively simple as we can see from these examples.

Temporarily Gaining root Privileges

In this example, user bob attempts to view the contents of the /etc/sudoers file, which is an action that normally requires privileged access. Without sudo, the command fails:

```
[bob@bigboy bob]$ more /etc/sudoers
/etc/sudoers: Permission denied
[bob@bigboy bob]$
```

Bob tries again using sudo and his regular user password and is successful:

```
[bob@bigboy bob]$ sudo more /etc/sudoers
Password:
...
...
[bob@bigboy bob]$
```

The details of configuring and installing sudo are covered in later sections.

Becoming root for a Complete Login Session

The `su` command allows a regular user to become the system's `root` user if they know the `root` password. A user with `sudo` rights to use the `su` command can become `root`, but they only need to know their own password, not that of `root` as seen here.

```
someuser@u-bigboy:~$ sudo su -
Password:
root@u-bigboy:~#
```

Some systems administrators will use `sudo` to grant `root` privileges to their own personal user account without the need to provide a password.

Later sections describe how to disable `sudo su` ability and also how to use `sudo` without password prompts.

Downloading and Installing the sudo Package

Fortunately the package is installed by default by RedHat/Fedora which eliminates the need to anything more in this regard. The visudo Command

The visudo command is a text editor that mimics the vi editor that is used to edit the /etc/sudoers configuration file. It is not recommended that you use any other editor to modify your sudo parameters because the sudoers file isn't located in the same directory on all versions of Linux. visudo uses the same commands as the vi text editor. The visudo command must run as user root and should have no arguments:

```
[root@aqua tmp]# visudo
```

The /etc/sudoers File

The /etc/sudoers file contains all the configuration and permission parameters needed for sudo to work. There are a number of guidelines that need to be followed when editing it with visudo.
General /etc/sudoers Guidelines

The /etc/sudoers file has the general format shown in Table 9-1.

Table 9-1 Format of the /etc/sudoers File

General sudoers File Record Format
`usernames/group servername = (usernames command can be run as) command`

There are some general guidelines when editing this file:

- Groups are the same as user groups and are differentiated from regular users by a % at the beginning. The Linux user group "users" would be represented by %users.
- You can have multiple usernames per line separated by commas.
- Multiple commands also can be separated by commas. Spaces are considered part of the command.
- The keyword ALL can mean all usernames, groups, commands and servers.
- If you run out of space on a line, you can end it with a back slash (\) and continue on the next line.
- sudo assumes that the sudoers file will be used network wide, and therefore offers the option to specify the names of servers which will be using it in the servername position in Table 9-1. In most cases, the file is used by only one server and the keyword ALL suffices for the server name.
- The NOPASSWD keyword provides access without prompting for your password.

Simple /etc/sudoers Examples

This section presents some simple examples of how to do many commonly required tasks using the sudo utility.

Granting All Access to Specific Users

You can grant users bob and bunny full access to all privileged commands, with this sudoers entry.

```
bob, bunny  ALL=(ALL) ALL
```

This is generally not a good idea because this allows bob and bunny to use the su command to grant

themselves permanent root privileges thereby bypassing the command logging features of sudo. The example on using aliases in the sudoers file shows how to eliminate this prob

Granting Access To Specific Users To Specific Files

This entry allows user peter and all the members of the group operator to gain access to all the program files in the /sbin and /usr/sbin directories, plus the privilege of running the command /usr/local/apps/check.pl. Notice how the trailing slash (/) is required to specify a directory location:

```
peter, %operator ALL= /sbin/, /usr/sbin, /usr/local/apps/check.pl
```

Notice also that the lack of any username entries within parentheses () after the = sign prevents the users from running the commands automatically masquerading as another user. This is explained further in the next example.

Granting Access to Specific Files as Another User

The sudo -u entry allows allows you to execute a command as if you were another user, but first you have to be granted this privilege in the sudoers file.

This feature can be convenient for programmers who sometimes need to kill processes related to projects they are working on. For example, programmer peter is on the team developing a financial package that runs a program called monthend as user accounts. From time to time the application fails, requiring "peter" to stop it with the /bin/kill, /usr/bin/kill or /usr/bin/pkill commands but only as user "accounts". The sudoers entry would look like this:

```
peter ALL=(accounts) /bin/kill, /usr/bin/kill /usr/bin/pkill
```

User peter is allowed to stop the monthend process with this command:

```
[peter@bigboy peter]# sudo -u accounts pkill monthend
```

Granting Access Without Needing Passwords

This example allows all users in the group operator to execute all the commands in the /sbin directory without the need for entering a password. This has the added advantage of being more convenient to the user:

```
%operator ALL= NOPASSWD: /sbin/
```

Using Aliases in the sudoers File

Sometimes you'll need to assign random groupings of users from various departments very similar sets of privileges. The sudoers file allows users to be grouped according to function with the group and then being assigned a nickname or alias which is used throughout the rest of the file. Groupings of commands can also be assigned aliases too.

In the next example, users peter, bob and bunny and all the users in the operator group are made part of the user alias ADMINS. All the command shell programs are then assigned to the command alias SHELLS. Users ADMINS are then denied the option of running any SHELLS commands and su:

```
Cmnd_Alias    SHELLS = /usr/bin/sh,  /usr/bin/csh, \
                       /usr/bin/ksh, /usr/local/bin/tcsh, \
                       /usr/bin/rsh, /usr/local/bin/zsh


User_Alias    ADMINS = peter, bob, bunny, %operator
ADMINS        ALL    = !/usr/bin/su, !SHELLS
```

This attempts to ensure that users don't permanently su to become root, or enter command shells that bypass sudo's command logging. It doesn't prevent them from copying the files to other locations to be run. The advantage of this is that it helps to create an audit trail, but the restrictions can be enforced only as part of the company's overall security policy.

Other Examples

You can view a comprehensive list of /etc/sudoers file options by issuing the command man sudoers.

Using syslog To Track All sudo Commands

All sudo commands are logged in the log file /var/log/messages which can be very helpful in determining how user error may have contributed to a problem. All the sudo log entries have the word sudo in them, so you can easily get a thread of commands used by using the grep command to selectively filter the output accordingly.

Here is sample output from a user bob failing to enter their correct sudo password when issuing a command, immediately followed by the successful execution of the command /bin/more sudoers.

```
[root@bigboy tmp]# grep sudo /var/log/messages
Nov 18 22:50:30 bigboy sudo(pam_unix)[26812]: authentication failure;
logname=bob uid=0 euid=0 tty=pts/0 ruser= rhost= user=bob
Nov 18 22:51:25 bigboy sudo: bob : TTY=pts/0 ; PWD=/etc ; USER=root ;
COMMAND=/bin/more sudoers
[root@bigboy tmp]#
```

Conclusion

It is important to know how to add users, not just so they can log in to our system. Most server based applications usually run via a dedicated unprivileged user account, for example the MySQL database application runs as user mysql and the Apache Web server application runs as user apache. These accounts aren't always created automatically, especially if the software is installed using TAR files.

Finally, the sudo utility provides a means of dispersing the responsibility of systems management to multiple users. You can even give some groups of users only partial access to privileged commands depending on their roles in the organization. This makes sudo a valuable part of any company's server administration and security policy.