Name That User

Exercise 14

What You Will Learn in this Lab:

- Add, Modify, and Remove Users
- Add, Modify, and Remove Groups
- Modify File Ownership

Details

First, let's review the type of accounts on a linux system

root account
This is also called superuser and would have complete and unfettered control of the system. A superuser can run any commands without any restriction. This user should be assumed as a system administrator.

System accounts
System accounts are those needed for the operation of system-specific components for example apache accounts and the sshd accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.

User accounts
User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Unix supports a concept of Group Account which logically groups a number of accounts. Every account would be a part of another group account. A Unix group plays important role in handling file permissions and process management.

Files
There are four main user administration files for users and groups:

- /etc/passwd − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- /etc/shadow − Holds the encrypted password of the corresponding account..
- /etc/group − This file contains the group information for each account.
- /etc/gshadow − This file contains secure group account information.

Page through each of the above files. Using a long listing, check the permissions on each of the above files.

Describe how the permissions on passwd and shadow differ. Be sure to include any differences in the ownership.

Commands
There are commands that allow us to add, modify, and delete users and groups. Since these are system management functions, you will need to preface each command with sudo.

Groups
Since users are placed into groups, we will start by looking at managing groups.

All of the default groups are system account specific groups. It is not recommended to use them for ordinary accounts.

The **groupadd, groupmod,** and **groupdel** commands allow us to manage groups. As we will see below in the user section, by default creating a user will also create a group with the same name by default. This means that you will only need to create a new group if you need specific users to share files among themselves.

Look at the **groupmod** man page. Add a group to your Pi called **fiends** (sic).

Display the file /etc/group. What is the UID of the fiends group?

Look at the **groupadd** man page. Change the name of the group **fiends** to **friends**.

Display the file /etc/group. What is the UID of the fiends group? Which users are in the sudo group?

Users
The **adduser, usermod,** and **deluser** commands allow us to manage users. Since we are on a Debian system, we will use adduser instead of the low-level command useradd. While both can add a new users, the adduser command does a number of steps for us automatically, such as creating the user's home directory. Similarly, we will be using deluser instead of userdel. We will start by looking how we can manually specify parameters with useradd, and will then see how it compares to adduser.

Before you begin, you should consider how you will name user accounts. Would you just use people's first names (which is fine until you have two Johns), first initial followed by last name (which is fine until you have both John and Jane Smith), or some other scheme (think about your SSID). In any case, try not to user excessively long names. To see how user are added to the system by default, type:

```
useradd -D
```

You should see something similar to:

```
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
```

If you looked at /etc/group, which lists the names of the groups on your system, you'll find number 100 as being assigned to **users**. This is a default, catch-all group of sorts, that includes all users.

`HOME=/home`
As you already know, a user added to the system will be given a directory to keep his or her files. This is created, by default, in /home.

`INACTIVE=-1`
refers to the number of days it takes for the system to lock you out when your account expires. As you can see here, you get locked out real soon!

`EXPIRE=`
refers to a date when your account is supposed to expire. In our example, there is no date. This is probably fine for regular employees, but if you hired some outside consultant to do work for you, you might want to his or her account to expire at the conclusion of business. The best way to handle temporary workers (and especially fired employees) is to delete the account entirely. But even the best administrators are often over-worked and may forget, so setting an expiration date will ensure that an account cannot be used.

`SHELL=/bin/bash`
refers to the shell that users will have to run commands on the system. Bash, or the Bourne Again Shell, is standard on Linux distributions. That doesn't mean that you as a system administrator can't install other shells on the system and make them the default (The Korn shell, for example).

`SKEL=/etc/skel`
means that the files located in /etc/skel will be copied to the users directory when you create the account.

You may want to add files to this directory or you may want to make changes to them. For example, you may want to modify .bashrc to make it easier for users. For example, at line 91 add the following alias to .bashrc:

```
alias ls-l='ls -l'
```

this would allow a user who has a typo and enters the long list command without a space to get the output they expected.

Consider the following command line (this is all one command line):

```
useradd -c "Mark Twain - AKA, Samuel Langhorne Clemens" -d
/home/mtwain  -m -k /etc/skel -g 100 -s /bin/bash mtwain
```

In it, we've added Mark Twain as a user to our system. The first option, -c, is a short comment that must come between quotation marks and helps identify our user in the /etc/passwd file. The second option, -d is where you want his personal directory to be created. The -m option creates the home directory and the -k option copies the files in the following directory (in our case, /etc/skel) there. The -g option puts the user in the group whose number follows. The -s option provides the user with a shell to use (in our case, Bash). Finally, the name the user will login with.

For the time being, Twain is without a password and therefore cannot log in. This is a bit more complicated. Most Linux distributions use a system of shadow passwords. These are encrypted so that a normal user can't look at the passwords in the system. The problem we have with useradd, is that it needs an already encrypted password to go with the **–p** option to work correctly. Let's say I wanted to give Twain the password 'h4ml3t'. If I added -p h4ml3t, the system would reject his login because it wasn't encrypted. If you looked at the /etc/shadow file where these encrypted passwords are stored, you'd find something like this next to a user's login name:

```
Q37spqpXAsl1Y
```

User jsmith may have the password 'mrT1bbs' but it would appear as something like

```
jsmith:F54spP9U4sl1X:12043:0:99999:7:::
```

in /etc/shadow. This is because the password has been created as an md5 hash, which is an encryption algorithm (an entry with an asterisk "*" indicates the account has been disabled). So, the bottom line here is that you have two options. An administrator that simply needs to add an account here and there can use the options above, minus the -p and then run the **passwd** command:

```
passwd mtwain
```

and provide a password. passwd will then create the md5 hash automatically.

Any entry in the /etc/passwd file that contains an "x" in the password field (the second field) means that the password is stored in the /etc/shadow file.

Now, let's compare the above Twain example with **adduser**:

```
adduser --gecos "Mark Twain - AKA, Samuel Langhorne Clemens" mtwain
```

In addition to doing everything that the sample useradd command does, this will also create a group called mtwain.

We can simply use the command with a username, and adduser will prompt us for various entries. Try, filling in information for all of the prompts:

```
adduser jsmith
```

Notice that the system creates the group, user, and home directory for us. It then prompts us for a password and optional user information. Be sure to note the password you assigned to this account, you will need it below.

Check the /etc/passwd and the /etc/group files for the jsmith entries. What are the UID and the GID for jsmith?

The **usermod** command allows us to modify a user's account. The most common use is to add the user to a supplementary group or to change the user's ID.

Read the usermod man page. Add the **friends** group as a supplementary group to the user jsmith. What command did you issue?

Check /etc/group to see if the user was added correctly. If it was, add the pi user to the same group.

Using usermod, change the user's login from jsmith to jdoe. Look at /etc/passwd and /etc/group. Notice that your command did not change both the user jsmith and the group jsmith to jdoe. What command would you also have to issue to change the group from jsmith to jdoe?

File Ownership

By default, a file is owned by the user who creates that file, and by that user's default group.

Display a long list of the files in your home directory and in /var/log. Who are the user and group owners of the files in your home directory? Who are the user and group owners of the file /var/log/auth.log ?

Using the **touch** command as the user pi (i.e., not with sudo), create an empty file in your home directory called demo. Check to make sure that the user owner is pi and the group owner is pi.

The **chown** command changes the user and group ownership of a file or directory. On our systems, only privileged users can issue the chown command, so you will need to use sudo with it.

Let's say that you wanted anyone in the **friends** group to have read and write access to the demo file. Using chown, change the group ownership of the demo file to the group **friends**, and change the permissions so the owner and the group have read and write access and anyone else has just read.

What commands did you issue to perform the above tasks?

Log into your pi as the user jdoe.

Type **pwd.** What is the full path for jdoe's home directory (Do not answer ~)? Is this what you expected? Why or why not?

Create an empty file in jdoe's home directory called **private.** Type:

`ls-l` (no space)

Did the command work? Would you expect this to work for the user pi? Why or why not?

By default, files in a home directory cannot be edited by anyone other than the owner. Change directories to /home/pi, and attempt to add some text to the /home/pi/demo file.

Were you able to add the text? If so, why was this allowed; if not what prevented you from added the text?

Log back in as the user pi.

Cleaning Up

The **deluser** command removes user accounts from the system. By default, deluser will remove the user without removing the home directory, the mail spool or any other files on the system owned by the user. Removing the home directory and mail spool can be achieved using the --remove-home option. If the --home option is given, deluser will only remove the user if the directory given to the --home option matches the user's real home directory.

Remove the jdoe account, but leave the directory in place.

Reflection

Answers to these questions must not be written on the lab worksheet, but typed on separate sheets of paper attached to the end of your lab worksheet or shared as a Google Doc. Your answers must be typed (not handwritten), and you will be graded on all aspects of your answer (correctness, use of proper terminology, readability, use of complete sentences only, etc.). In general you are expected to write at least one or two paragraphs in answer to each question.

Assume that you are setting up a new server for an organization that has three departments: Sales, ITS, and Accounting. You will keep the system restriction that a casual user may not issue a chown. How would you set up the users' default and supplementary groups to allow users in a given department the ability to share files in their home directory with other users in their department, but still restrict access to the users not in their department.

This lab was based on material found at https://www.tutorialspoint.com/unix/unix-user-administration.htm and https://www.linux.org/threads/users-on-a-system.4228/