Unix users and groups

1 Overview

This exercise introduces management of users and groups on a Unix system. The lab includes the following objectives:

- Add users to a shared system.
- Define a group on the system, and assign users to that group.
- Observe how user and group IDs can affect access to files.
- Observe a limitation of discretionary access controls.
- Grant a user *sudo* or *superuser* privileges.

This lab does not provide in-depth coverage of users, groups and Unix permissions. It introduces the concepts and provides hands-on experience. Use the Unix help facilities, e.g., `useradd -h` to view detailed options for different commands covered in this lab.

1.1 Background

This is an introductory lab that only requires a bit of familiarity with the Unix command line.

2 Lab Environment

This lab runs in the Labtainer framework, available at http://nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer users
```

A link to this lab manual will be displayed.

3 Environment

While interfaces for adding and managing users may vary between installations, the concept of Unix users and groups are the same across all Unix and Linux systems. In this lab you will manage a CentOS7 system using the *useradd* command.

When the lab starts, two virtual terminals appear, each with a standard Unix login prompt. Both terminals connect to the same computer, called *shared*. Initially, there is one user named *admin* defined, with a password of *password123*.

Login to one of the terminals as admin and use `sudo su` to get a root shell. Use that root shell to manage users for the lab. Use the other terminal to test those users per the lab directions below.

4 Tasks

4.1 Add user Bob

Use this command to add a user with id `bob`:

```
useradd -m bob
```

The `-m` option causes a home directory to be created for the user at `/home/bob`. Go to the other login terminal and try to login as the new user bob. Oh, that's right, the user does not yet have a password. Passwords are assigned and changed using the `passwd` program. Use:

```
man passwd
```

to learn about options for the `passwd program`. For this lab, we'd like to assign bob a password so that bob can login, but we want the password to be expired so that bob is forced to change it. Use the `passwd` program to assign bob a password, and then run the `passwrd` program to cause bob's password to be expired. Then login as bob and confirm that bob is required to change the password. Do that and make note of the password for later. (For this lab it is **OK** to write down passwords!)

Then use `exit` to log bob out of the terminal.

4.2 Add user Mary

Add a user with id `mary` just like you added bob. Before you try to login as Mary, take a look at a file that Mary will need to access. Use this command:

```
ls -l /shared_stuff/tarts.txt
```

Note who owns the file, and the permitted access modes for the owner, the group and other.

```
-rw-rw---- 1 frank bakers 8 Apr 29 23:48 /shared_stuff/tarts.txt
```

A brief tutorial on Unix file permissions can be viewed at: `https://mason.gmu.edu/~montecin/UNIXpermiss.htm`. For this file, the owner has read and write permission, as do members of the group. Users other than the owner or members of the group have no access to the file. The owner is `frank` and the group is `bakers`. The permissions indicate that only members of the bakers group should be able to read or write the file.

We want Mary to be able to access this file because she is a baker. So, add Mary to the bakers group. This is done with the `usermod` command:

```
usermod -a -G bakers mary
```

Now login to the terminal as mary. Then use the `id` command to confirm that mary is in the bakers group. Then confirm at Mary can view the file:

```
cat /shared_stuff/tarts.txt
```

Now run one of the bakers' application programs:

```
eggcheck tarts.txt
```

When the eggcheck program runs, it runs as mary with her permissions. This lets the program read the file.

Use the `id` command again. Note how the user is mary and the group is mary. While mary is a member of both the mary group and the bakers group, her current id reflects the mary group. Use this command to create a new file as mary:

```
touch newfile.txt
```

Use `ls -l` to view ownership of the file. Then change Mary's current group by using:

```
newgrp bakers
```

and use `ls -l` again. Many things can affect the permissions of a file, including the current group of the user creating the file.

4.3 Re-login as bob

Use `exit` to exit the mary session and login as bob again. Try to view the tarts.txt file:

```
cat /shared_stuff/tarts.txt
```

Since Bob is not a member of the bakers group, Bob is not able to access the file. This type of control over access to files is called *discretionary access control* because it relies on the discretion of the members of the bakers group. For example, at their discretion, a member of the bakers group could create a copy of the tarts.txt file that is visible to users who are not in the bakers group. However, it is not only the discretion of the users, but also the discretion of the *programs* executed by those users. Recall Mary ran the `eggcheck` program. That program happens to creates a temporary copy of the file that it reads, and places that copy in `/tmp/tmpfile.txt`. Try to cat that file as bob. While none of the bakers used their discretion to make that recipe available to everyone, it happened anyway as a side effect of the implementation of an application used by the bakers.

This is the nature of discretionary file controls such as those in Unix. They rely on the user knowing what the programs are actually doing. And that is not always the case.

4.4 Add a privileged user

Add a new user to your system with an id of `lisa`. Lisa will help administer the system and requires the ability to perform privileged functions, sometimes referred to as *sudo*, short for "Superuser do".

In modern Unix systems, superuser privileges are assigned to members of groups defined in the `/etc/sudoers` file. In older CentOS systems, membership in the group *wheel* provided superuser privileges. That group no longer has special significance. View the `/etc/sudoers` file to see that in our installation, membership in the `admin` group is what provides this privilege. Add lisa to the admin group just as you added mary to the bakers group.

Then login as lisa and perform the privileged function of removing a user. In this case, terminate the bob user with this command:

```
sudo userdel bob
```

Then try to login as bob to confirm the user was deleted.

5 Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab
```

When you stop the lab, the system will display a path to the zipped lab results on your Linux system. Provide that file to your instructor, e.g., via the Sakai site.