

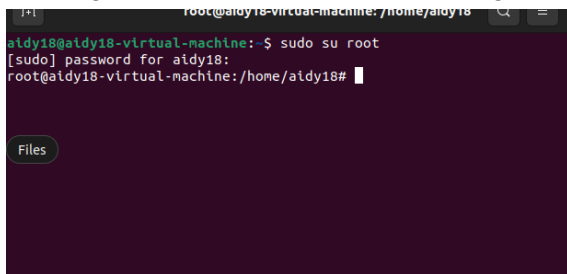
Github: [https://github.com/Aidy18/csec\\_home\\_lab](https://github.com/Aidy18/csec_home_lab)

## Command Line and Security Basics

- This activity was done to demonstrate and understand some of the tools in the Linux command line, as well as demonstrating some good (and bad) security practices.
- Everything was done using a virtual machine (VM) running Ubuntu 22.04.5 (Desktop)

### Users

After booting the VM and updating its packages, I first open a root shell using `sudo su root`. This logs the user in as root and changes the prompt to look like the one shown below.

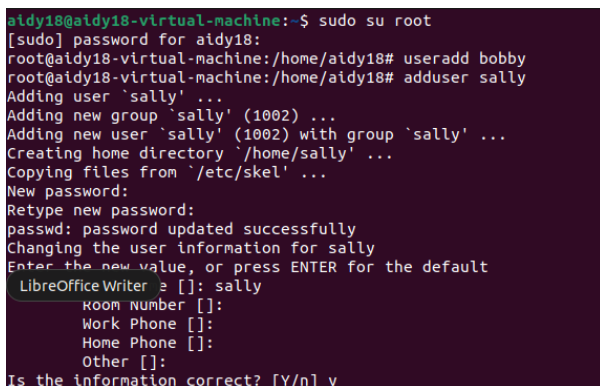


```
root@aidy18-virtual-machine: /home/aidy18
aidy18@aidy18-virtual-machine:~$ sudo su root
[sudo] password for aidy18:
root@aidy18-virtual-machine: /home/aidy18#
```

This is a shell that allows you to execute commands regardless of permission, since it is the root user. It is never safe to be logged in as root, since if another user has this shell, they can gain access to data they aren't supposed to or modify whatever files they want.

While logged in as root, I then make two users in two different ways:

A user named bobby is made with `useradd bobby`; then another user sally is made with `adduser sally` below



```
aidy18@aidy18-virtual-machine:~$ sudo su root
[sudo] password for aidy18:
root@aidy18-virtual-machine: /home/aidy18# useradd bobby
root@aidy18-virtual-machine: /home/aidy18# adduser sally
Adding user `sally' ...
Adding new group `sally' (1002) ...
Adding new user `sally' (1002) with group `sally' ...
Creating home directory `/home/sally' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for sally
Enter the new value, or press ENTER for the default
  LibreOffice Writer & []: sally
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

We see there's a clear difference between the two methods. With `adduser` you can add more details, but more importantly a password to the user upon creation. `useradd` however simply creates the user and doesn't specify a password, you'd have to use a separate command to set the password (like `sudo passwd`).

From root, I use `sudo su sally` to login as the user sally. The prompt changes, and I test `useradd` to see if it will add a user earl. It won't work since sally is not in the group that has permission to modify the `passwd` file, where all of the user information is stored.

```
sally@aidy18-virtual-machine:/home/aidy18$ useradd earl
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

I exit back to the original user and proceed to try `userdel` to delete a nonexistent user earl, and the already existing bobby as seen below.

```
aidy18@aidy18-virtual-machine:~$ sudo userdel earl
[sudo] password for aidy18:
userdel: user 'earl' does not exist
aidy18@aidy18-virtual-machine:~$ sudo userdel bobby
aidy18@aidy18-virtual-machine:~$ sudo su bobby
su: user bobby does not exist or the user entry does not contain all the required fields
```

## Groups

I now shift focus to groups in Linux. I start with using `id` to see what groups the original user is a part of, all of which are shown below.

```
aidy18@aidy18-virtual-machine:~$ id
uid=1000(aidy18) gid=1000(aidy18) groups=1000(aidy18),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),122(lpadmin),135(lxd),136(sambashare)
```

Seeing as this user has sudo privileges, I proceed to use those privileges to make the user sally a sudoer with `sudo usermod -a -G sudo sally`. Then, I login with sally and check the groups to show that it worked.

```
aidy18@aidy18-virtual-machine:~$ sudo usermod -a -G sudo sally
aidy18@aidy18-virtual-machine:~$ sudo su sally
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

sally@aidy18-virtual-machine:/home/aidy18$ id
uid=1002(sally) gid=1002(sally) groups=1002(sally),27(sudo)
sally@aidy18-virtual-machine:/home/aidy18$
```

From there, I try to now add the user bobby2 as sally, who now has the privileges to do so successfully.

```
sally@aidy18-virtual-machine:/home/aidy18$ sudo useradd bobby2
[sudo] password for sally:
Sorry, try again.
[sudo] password for sally:
sally@aidy18-virtual-machine:/home/aidy18$ users
aidy18 aidy18
Help sally@aidy18-virtual-machine:/home/aidy18$ sudo su bobby2
```

I now try to make a new group called `cybersec` using `sudo groupadd cybersec`. I then add sally to the group and use `grep` to search the `/etc/group` file to see what groups sally belongs

to, which now includes the newly made cybersec group, matching wherever sally shows up.

```
aidy18@aidy18-virtual-machine:~$ sudo groupadd cybersec
aidy18@aidy18-virtual-machine:~$ sudo usermod -a -G cybersec sally
aidy18@aidy18-virtual-machine:~$ grep sally /etc/group
sudo:x:27:aidy18,sally
sally:x:1002:
cybersec:x:1004:sally
```

## Permissions and Access Control Lists

To understand permissions and the fine-grain access control list (fACL) of files and directories, I start by making a folder lab1 and checking its permissions with `getfacl`

```
aidy18@aidy18-virtual-machine:~$ mkdir lab1
aidy18@aidy18-virtual-machine:~$ getfacl lab1
# file: lab1
# owner: aidy18
# group: aidy18
user::rwx
group::rwx
other::r-x
```

The owner of the file is aidy18, the group owner being the same. Both the group and file owner have read, write, and execute permissions for the folder, so they can see the contents, add contents, and enter the folder. Other users have only read and execute permissions, so they can see the contents and enter the folder, but cannot add to it.

I change the directory to the new folder and make a bash script to display “hello world”.

```
#!/bin/bash

echo "Hello world!"
```

This is saved and made executable with `chmod u+x helloWorld.bash` and it runs as expected.

```
aidy18@aidy18-virtual-machine:~/lab1$ getfacl helloWorld.bash
# file: helloWorld.bash
# owner: aidy18
# group: aidy18
user::rw-
group::rw-
other::r--

aidy18@aidy18-virtual-machine:~/lab1$ chmod u+x helloWorld.bash
aidy18@aidy18-virtual-machine:~/lab1$ ./helloWorld.bash
Hello world!
aidy18@aidy18-virtual-machine:~/lab1$
```

I then check the permissions of the file with `ls -al helloWorld.bash`

It can be faintly seen that the file owner has read, write and execute permissions, the group owner has read and write permissions, and other users have only read permissions. I then change the group permissions to have write and execute permissions with `chmod`.

```

aidy18@aidy18-virtual-machine:~/lab1$ ls -la helloWorld
ls: cannot access 'helloWorld': No such file or directory
aidy18@aidy18-virtual-machine:~/lab1$ ls -la helloWorld.bash
-r-- 1 aidy18 aidy18 33 Oct  2 11:32 helloWorld.bash
aidy18@aidy18-virtual-machine:~/lab1$ chmod g+wx helloWorld.bash

```

Following this, I get the access control list of the file with `getfacl`, giving the updated permissions

```

aidy18@aidy18-virtual-machine:~/lab1$ getfacl helloWorld.bash
# file: helloWorld.bash
# owner: aidy18
# group: aidy18
user::rwx
group::rwx
other::r--

```

I then set the user sally specifically to have read and write permissions on the file `helloWorld`. When logged in as sally, we see that we can use `cat` to read the file. And if we open a text editor to make changes, we can do something like change the output as shown.

```

aidy18@aidy18-virtual-machine:~/lab1$ setfacl -m u:sally:rw helloWorld.bash
aidy18@aidy18-virtual-machine:~/lab1$ sudo su sally
sally@aidy18-virtual-machine:/home/aidy18/lab1$ cat helloWorld.bash
#!/bin/bash

echo "Hello world!"
sally@aidy18-virtual-machine:/home/aidy18/lab1$ nano helloWorld.bash
sally@aidy18-virtual-machine:/home/aidy18/lab1$ cat helloWorld.bash
#!/bin/bash

echo "Hello world! Sally was here"
sally@aidy18-virtual-machine:/home/aidy18/lab1$ exit
exit
aidy18@aidy18-virtual-machine:~/lab1$ cat helloWorld.bash
#!/bin/bash

echo "Hello world! Sally was here"

```