The University of Newcastle
School of Information and Physical Sciences

SENG2250/6250 System and Network Security
Week 8 Lab

Part 1: Operating System Security (Ubuntu/Linux Practices)

The objectives of this lab are twofold:

1) Explore the commands under Ubuntu.
2) Exercise the access control and file permission operations in the Ubuntu environment.

Lab Environment and Notes:

1) Log in to any Ubuntu VM on the virtual platform at https://cybersec-vra04.newcastle.edu.au/. Refer to the VMware Online Platform Instructions from the Week 6 lab if you have forgotten how.
2) Note: You don't need to request the VM if you already did so in Week 6. You can go directly to the 'Deployment' tab to access your VMs.
3) Username: Student

   Password: $tud3nt

The output of your environment might be slightly different from the screenshot of this specification. For example, the *i*node value of a file and the file details (e.g., time, shell info) may differ on your machine.

Ubuntu Commands: Some of the frequent commands are given below.

More Linux/Unix commands: https://www.unixtutorial.org/commands

Command	Description	Example
.	Current directory	
..	Parent directory	
~	Home directory of a user	/home/user1
cd	Change to a directory	cd ..
find	Find a file by using filters like name	find . -name "filea*"
touch	Create an empty file (example)	touch file1
mkdir	Create a directory	mkdir newdirectory
mv	Move or rename file	mv file1 file1-new
cp	Copy file	cp file1 file2
rm	Remove file	rm file

ls	List files and directories	ls
sudo	Run command as a super user	sudo rm file
grep	Search keywords in a file	grep -n keyword file
pwd	Display current working dir path	pwd
cat	Display a file	cat file
more	Display a file in pages	more file
date	Display current time	date
kill	Terminate a program	kill PID
man	Display manual of a command	man find
who	list current users	who
>	Redirect standard output to file	command > file
<	Redirect standard input from file	command < file
*	match any number of characters	
?	match one character	

Linux file permission system: The traditional Unix security model is based on the discretionary access control (DAC) model, which enables users to configure who can access the resources that they "own". Each user can control which other users can access the files that they create.

This enables users to grant permissions, without involving a system admin. This is the type of security that has traditionally been built into most consumer OSs such as Windows and Unix.

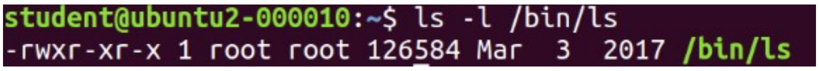
Unix file permissions use an abbreviated (simplified) form of access control list (ACL). A (full) ACL involves attaching a list of every subject and what they can do to each file (this is how Windows manages file access). For example, a file may have this ACL: "Joe can read, Frank can write, Alice can read, and Eve can read". Unix simplifies permissions by only defining rules for these three kinds of subjects:

- User: who owns the file (u)
- Group: users belonging to the file's ownership group (g)
- Other: Other users/ everyone else (o)

Example 1:

- Step1: Open Linux terminal and type the following command to see the file permission of any Linux executable file.
- Step2: Use the *ls* command to display the permissions for a file (the details of the *ls* executable Program itself): *ls -l /bin/ls*

The "-l" flag instructs ls to provide this detailed output.



```
student@ubuntu2-000010:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 126584 Mar  3  2017 /bin/ls
```

The first block in the output line shows different permissions. They are listed in the table below:

rwx	r-x	r-x
Owners Privileges	Groups Privileges	Others' Privileges
Owner can read, write, and execute	Group members can read and execute	Anyone can read and execute

The meaning of these letters is fairly self-evident, but does change meaning slightly depending on whether it refers to a normal file or a directory (which is really just a special kind of file).

For a regular file	**For a directory**
• r: Read the contents of the file • w: Change the contents of the file • x: Execute the file as a process (The first few bytes describe what type of executable it is, a program or a script)	• r: See what files are in the directory • w: Add, rename, or delete names from the directory • x: 'stat' the file (view the file owners and sizes, cd into the directory, and access files within) • t (instead of x), AKA the "sticky bit": write is not enough to delete a file from the directory, in this case you also need to own the file

The rest of the output from ls describes how many names/hard links the file has, who owns the file (user and group associated with the file), the file size in bytes, the last access date, and finally the path and name of the file.

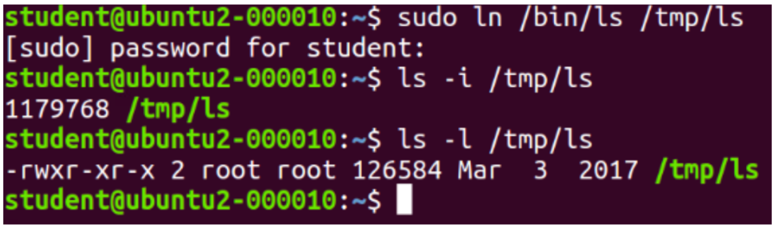
1	root	root	126584	Mar 3 2017	/bin/ls
The file has this many names in the hard drive	Root owns the file	File belongs to root file group	File size	Date last modified	File path

Q1: Check the file permission for **/bin/cat**

Describe Owner Privileges:	
Describe Group Privileges:	
Describe Others' Privileges:	
File Belongs to:	
Last modification date:	

Example2: Create a hard link to ls program

sudo ln /bin/ls /tmp/ls



```
student@ubuntu2-000010:~$ sudo ln /bin/ls /tmp/ls
[sudo] password for student:
student@ubuntu2-000010:~$ ls -i /tmp/ls
1179768 /tmp/ls
student@ubuntu2-000010:~$ ls -l /tmp/ls
-rwxr-xr-x 2 root root 126584 Mar  3  2017 /tmp/ls
student@ubuntu2-000010:~$ 
```

The inodes are identical and it has two hard links. Which implies that, data can now be accessed using two different names /tmp/ls and /bin/ls. If the /tmp/ls file was edited, the /bin/ls command would also change.

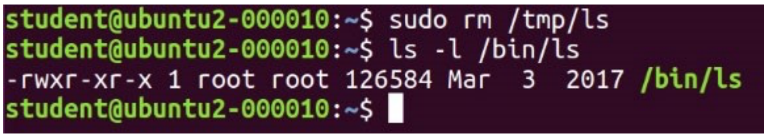
Q2: Why inodes are used?

Q3: What would be the permission value for an executable file?

Example 3: Remove a hard link

Deleting one of the names simply decrements the link counter. Only when the counter reaches 0, the inode is actually removed.

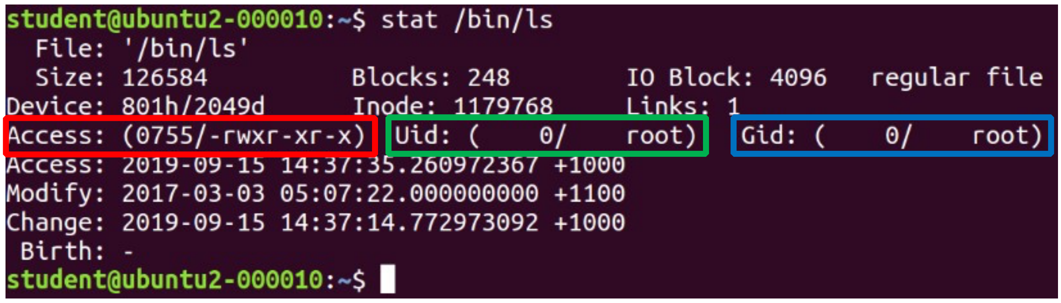
Remove a hard link is the same as removing a file: sudo rm /tmp/ls



```
student@ubuntu2-000010:~$ sudo rm /tmp/ls
student@ubuntu2-000010:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 126584 Mar  3  2017 /bin/ls
student@ubuntu2-000010:~$ 
```

Deleting the hard link decrements the counter. Now the counter is 1.

Example 4: stat command shows further information from the inode



```
student@ubuntu2-000010:~$ stat /bin/ls
  File: '/bin/ls'
  Size: 126584          Blocks: 248        IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1179768     Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-09-15 14:37:35.260972367 +1000
Modify: 2017-03-03 05:07:22.000000000 +1100
Change: 2019-09-15 14:37:14.772973092 +1000
 Birth: -
student@ubuntu2-000010:~$ 
```

Red Block represents the file permissions. The green block represents the user who owns the file (UID=0, meaning root owns the file). And finally, the blue block represents files user group (GID=0).

Each of the other three octets simply represents the binary for rwx, each represented as a 0 or a 1. The permission value calculation is illustrated below:

D?	Owner			Group			Others			Value
-	r	w	x	r	-	x	r	-	x	
0	1	1	1	1	0	1	1	0	1	755(D)
EXAMPLE										
-	r	w	-	r	w	-	r	w	-	
0	1	1	0	1	1	0	1	1	0	666(D)

Q4: Create and delete one more inode for /etc/network/interfaces. Also view inode permission for the file.

File Permissions: The **chmod** command can be used to set permissions on a file. **chmod** can set permissions based on absolute octal values, or relative changes.

Q5: Create a file and executable permission for all (use **touch** and **chmod**).

Q6: Make a directory containing multiple files and set executable permission for all files. (Hint: use 'man chmod')

Create new group

- Step 1: Create a Group named students: *sudo groupadd students*
- Step 2: Adding new user (dummysa) to the group students:
  sudo useradd -g students dummysa
- Step 3: Set password for dummysa: *sudo passwd dummysa*
- Step 4: to view the group: *cat /etc/group*
- Step 5: Repeat the same process to create a group named teachers.
- Step 6: Repeat the same process to create two more users named "dummysb" and "dummyta". Add "dummysb" to students group and "dummyta" to teachers group.

Create a new user that belongs to the root group

- Step 1: Type the following instruction to create a user named 'dummy'. Note that we have assigned a UID (-ou) and GID (-g) as 0, to make it a member of root group.

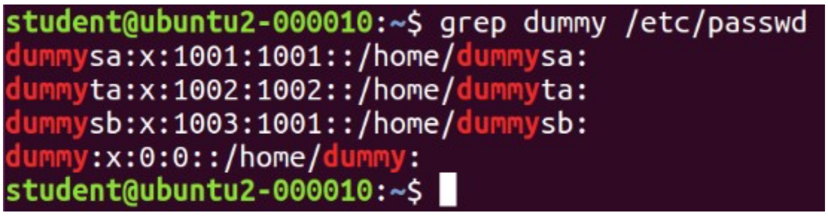
  sudo useradd -ou 0 -g 0 dummy

- Step 2: Set password to the new user. *sudo passwd dummy*

- Step 3: Check if the user is created or not.

  cat /etc/passwd {Too clumsy to read the details}

  grep dummy /etc/passwd { Read specific information about a user, in this case its dummy}



```
student@ubuntu2-000010:~$ grep dummy /etc/passwd
dummysa:x:1001:1001::/home/dummysa:
dummyta:x:1002:1002::/home/dummyta:
dummysb:x:1003:1001::/home/dummysb:
dummy:x:0:0::/home/dummy:
student@ubuntu2-000010:~$
```

The following commands (any one of three) can show a list of users in '/etc/passwd' file.

awk -F':' '{ print $1}' /etc/passwd

cut -d: -f1 /etc/passwd

compgen –u

Q7: Find and print the record of the user "student", explain the printed information.

Q8: Where are the users' password entries? Find and print the record of the user student's password. Explain the printed information.

Grant root privileged to a user

- Step 1: if it's an existing user change the UID and GID to 0. Edit '/etc/passwd' file. In our case we have already assigned them to 0.
- Step 2: Use the following instruction

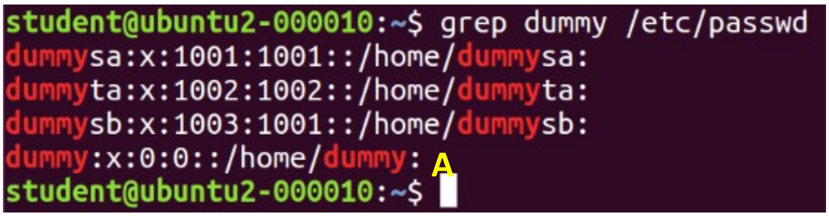
  sudo usermod -a -G root dummy

Delete a user

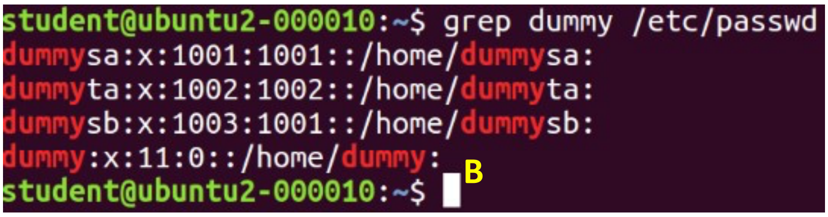
- Step 1: Edit the entry in '/etc/passwd' file. (Change from A→B)

  i). Use gedit to edit the file

  sudo gedit /etc/passwd



```
student@ubuntu2-000010:~$ grep dummy /etc/passwd
dummysa:x:1001:1001::/home/dummysa:
dummyta:x:1002:1002::/home/dummyta:
dummysb:x:1003:1001::/home/dummysb:
dummy:x:0:0::/home/dummy:
student@ubuntu2-000010:~$
```



```
student@ubuntu2-000010:~$ grep dummy /etc/passwd
dummysa:x:1001:1001::/home/dummysa:
dummyta:x:1002:1002::/home/dummyta:
dummysb:x:1003:1001::/home/dummysb:
dummy:x:11:0::/home/dummy:
student@ubuntu2-000010:~$
```

- Step 2: *sudo userdel dummy*

Change file group and ownership

- Step 1: Create a file being Root

  touch t1.txt

  ls -l t1.txt
- Step 2: Changing the owner and group to "dummysb" and students respectively.

  sudo chown dummysb:students t1.txt

Q9. Who can write to this file, all of dummysa, dummysb and dummyta?

Q10. Who can change the ownership of this file? And why?

Q11. Will you be able to create file being a normal user (non-root)? And why?

Apt install acl

How to use Linux ACL

Install acl tool

sudo apt install acl

User getfacl and setfacl commands.

- ***getfacl***: Get ACL on file or directory. Format: *getfacl file_or_dir_name*
- ***setfacl*** - *Set ACL on file or directory. Options: -m* modify; *-x* delete; *d* default (for directory).

  For example:

  setfacl –m d:u:user1:rw my_directory

  setfacl –x u:user1 my_file

  setfacl –m g:staff:rx my_db

Example 5:

- Step 1: To display the ACL of a file: *getfacl t1.txt*
- Step 2: Use the ACL to give user “dummysb” the right to read, write and execute t1.txt file.

  setfacl –m user:dummysb:rwx t1.txt

- Step 3: Check the permission using getfacl.

Q12: Create a new file and new user (non-root). Assign the file execution permission to this user only. Use chown, setfacl, getfacl