VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

CYBR371: System and Network Security
2024– T1

Arman Khouzani, Mohammad Nekooei

Lab 1: Linux Account Management and ACLs

Submission Deadline: ***23:59:00 (NZST) on Sunday, 17 March 2024***

EXAMPLE SOLUTIONS + GRADING RUBRIC

Instructions: Sign in into the NetLab (https://netlab.ecs.vuw.ac.nz/) with the provided credentials and complete the following labs:

- Lab 1: Linux Account Management
- Lab 2: Linux Access Control List

1 Part 1 - Linux Account Management

Complete the lab "Linux Account Management" and answer the following questions:

Q 1.1 [4 points] What is the octal or string representation of the following permissions?

```
rwxrw-r-t       =
r-S-wx--x       =
rwxr-xr--       =
r-Sr-sr-x       =
432             =
3532            =
6713            =
1530            =
```

Solution:

```
rwxrw-r-t       = 1765
r-S-wx--x       = 4431
rwxr-xr--       = 0754
r-Sr-sr-x       = 6455
432             = r---wx-w-
3532            = r-x-ws-wT
6713            = rws--s-wx
1530            = r-x-wx--T
```

Rubric: 0.5 point for each fully correct entry (no partial points), for a total of 4 points.

Q 1.2 [2 points] If the `umask` value for a user is `035`, what are the default file and directory permissions set for the user? Write the permissions and how they were calculated.

Solution:

- 0: take away no rights away from the owner user
- 3: take away write (2) and execute (1) permissions from the owner group
- 5: take away read (4) and execute (1) permissions from the other.

Hence, for (newly created) directories, whose default permissions are `777`, or `wrx wrx wrx`, we get to **`rwx r-- -w-`** (or **`742`** in short)

For (the newly created) files, starting from the default permissions of `666`, or `rw- rw- rw-`, we get to **`rw- r-- -w-`** (or **`642`** in short).

Rubric: 0.5 point for each correct value and 0.5 point for each correct explanation of derivation, for a total of 2 points.

Q 1.3 [2 points] If the default permissions given to directories that the user `xyz` creates are `rwxr-x---`, what are the default permissions set for the files created by the user? Write the permissions and how they were calculated.

Solution: For new directories, starting from `777`, or `rwx rwx rwx`, we have arrived at `rwx r-x ---`, so the umask has taken away:

- nothing from the owner user
- write permission from the group
- read, write, and execute from the others.

So, for new files, starting from `666`, or `rw- rw- rw-`, we get to:

- `rw-` for the owner user
- `r--` for the group owner
- `---` for others

That is, **`rw- r-- ---`** (or **`640`** in octal).

Rubric: 1 point correct permissions and 1 point explanation (any cl

Q 1.4 Find all the executables that have **`SUID`** set. Hint: you can use the **`find`** command with the right parameters.

(a) [2 points] Provide the command that you used to find these programmes.

Solution:

```
sudo find / -perm /4000 -type f
```

or if you want more information (like the owner user and groups and the rest of permissions on the file):

```
sudo find / -perm /4000 -type f -exec ls -la {} \;
```

if you want to not see any errors (diverting them into oblivion!):

```
sudo find / -perm /4000 -type f -exec ls -la {} \; 2>/dev/null
```

Rubric: 2 points for any correct entry. Note that `-perm /4000` and `-perm -4000` has the same effect here (`/mode` indicated any of the permission bits specified in `mode` and `-perm -mode` indicates at least the permission bits specified in `mode`; but, since here there is only 1 bit in 4000, these two will have the same effect. However, using `-perm 4000` (no leading slash or dash for 4000) will be wrong, since that will search for the exact permission of 4000.

(b) [4 points] List all the programmes you found. For 3 of them (of your choice), provide a brief description why it needs to be a SUID programme.

Solution: With the first command, on the Ubunbtu machine, here is the output:

```
student@Ubuntu:~$ sudo find / -perm /4000 -type f
find: `/proc/2629/task/2629/fd/5': No such file or directory
find: `/proc/2629/task/2629/fdinfo/5': No such file or directory
find: `/proc/2629/fd/5': No such file or directory
find: `/proc/2629/fdinfo/5': No such file or directory
find: `/home/student/.gvfs': Permission denied
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/pt_chown
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/telnetlogin
/usr/sbin/pppd
/usr/sbin/vmware-user-suid-wrapper
/usr/sbin/uuidd
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/X
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/arping
/usr/bin/sudoedit
/usr/bin/sudo
/usr/bin/traceroute6.iputils
/usr/bin/newgrp
/usr/bin/lppasswd
/usr/bin/at
/usr/bin/mtr
/usr/bin/chsh
/bin/ping6
/bin/mount
/bin/umount
/bin/su
/bin/ping
/bin/fusermount
```

and with the last command, the output is the following:

```
student@Ubuntu:~$ sudo find / -perm /4000 -type f -exec ls -al {} \; 2>/dev/null
-rwsr-xr-x 1 root root 9760 Sep 11  2013 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root root 9728 Jul 29  2014 /usr/lib/pt_chown
-rwsr-xr-- 1 root messagebus 316824 Jul  3  2014 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 5564 Dec 13  2011 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 248056 Apr  2  2012 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root telnetd 9872 Mar  6  2010 /usr/lib/telnetlogin
-rwsr-xr-- 1 root dip 273272 Feb  4  2011 /usr/sbin/pppd
-rwsr-xr-x 1 root root 9724 Jan 24  2012 /usr/sbin/vmware-user-suid-wrapper
-rwsr-sr-x 1 libuuid libuuid 17976 Jun 17  2014 /usr/sbin/uuidd
-rwsr-xr-x 1 root root 18104 Sep 11  2013 /usr/bin/pkexec
-rwsr-xr-x 1 root root 57956 Sep 12  2012 /usr/bin/gpasswd
-rwsr-sr-x 1 root root 9524 May 26  2014 /usr/bin/X
-rwsr-xr-x 1 root root 40292 Sep 12  2012 /usr/bin/chfn
-rwsr-xr-x 1 root root 41284 Sep 12  2012 /usr/bin/passwd
-rwsr-xr-x 1 root root 13860 Nov  8  2011 /usr/bin/arping
-rwsr-xr-x 1 root root 69708 Mar 11  2014 /usr/bin/sudoedit
-rwsr-xr-x 1 root root 69708 Mar 11  2014 /usr/bin/sudo
-rwsr-xr-x 1 root root 14012 Nov  8  2011 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 30896 Sep 12  2012 /usr/bin/newgrp
-rwsr-xr-x 1 root lpadmin 9768 Jul 18  2014 /usr/bin/lppasswd
-rwsr-sr-x 1 daemon daemon 42800 Oct 25  2011 /usr/bin/at
-rwsr-xr-x 1 root root 56208 Jul 28  2011 /usr/bin/mtr
-rwsr-xr-x 1 root root 31748 Sep 12  2012 /usr/bin/chsh
-rwsr-xr-x 1 root root 39116 Nov  8  2011 /bin/ping6
-rwsr-xr-x 1 root root 88760 Jun 17  2014 /bin/mount
-rwsr-xr-x 1 root root 67720 Jun 17  2014 /bin/umount
-rwsr-xr-x 1 root root 31116 Sep 12  2012 /bin/su
-rwsr-xr-x 1 root root 34740 Nov  8  2011 /bin/ping
-rwsr-xr-x 1 root root 26252 Mar  2  2012 /bin/fusermount
```

- **`passwd`**: allows a user to change their own password. For that, this needs ability to edit the `/etc/shadow` (and/or the `/etc/passwd`) file, which has to have no permission for others (otherwise anyone would be able to read or modify the hashed passwords of all other users, or delete another user, or add a new user with a given password and make it a member of sudo, etc). Similar argument applies to **`chsh`** and **`chfn`**, which respectively allows a user to change their own login shell, and their full name information, both of which requires the user to be able to edit the contents of `/etc/passwd`, which has to have no write permission for others, otherwise anyone could delete an existing user, change their login shell, etc.

- **`mount`** and **`umount`**: these command are used for mounting (attaching a filesystem to a directory in the existing filesystem hierarchy so that it becomes accessible to the system, e.g. a hard-drive, or a usb-stick) and removing it, respectively. These commands usually need root access because they affect the global namespace of the system, involve interacting with hardware devices, etc., which has serious security implications. However, in cases like using removable storage (i.e., letting a user mount their usb-storage) or in multi-user environments with restricted permissions, non-root users may need to perform these operations.

- **`su`**: for switching user, i.e., log in as a different user, e.g. as root, etc. This needs to be setuid, because it can be used to launch a shell with a different effective user id than the current user id of a user, a privileged operation. Moreover, this programme must ask for authentication before allowing the switch, but this means it should have access to the `/etc/shadow` file (to verify that the entered password is correct), which (as was the case in **`passwd`**) needs root privilege. Similar arguments apply to **`sudo`** and **`newgrp`**, as they both changing your privileges (which may require authentication).

Rubric: 1 point for the list. 3 points for explanations, 1 point per each. 4 points in total.

2 Part 2 - Linux Access Control List (ACL)

Complete the "Linux Access Control List" lab on netlab and answer the following questions. Please note that the questions below are dependent on the sequence of the lab instructions and must be followed and answered step by step as they appear in the "Linux Access Control List" lab document.

Q 2.1 [4 points] Write the command(s) you used to add the users with their associated provided information.

Solution:

```
useradd -m -s /bin/bash -g sudo cybr371
useradd -m -s /bin/bash -g sudo ben
useradd -m -s /bin/bash -g david david
useradd -m -s /bin/bash -g mary mary
useradd -m -s /bin/bash -g masood masood
```

instead of the last three commands, you can also use:

```
useradd -m -s /bin/bash -g david
useradd -m -s /bin/bash -g mary
useradd -m -s /bin/bash -g masood
```

Also, each of these also have a long format as well (which of course does the same thing, just check the manual of **`useradd`** command), e.g.:

```
useradd --create-home --shell /bin/bash --groups sudo cybr371
```

You can also use `-d` to set the home directory. If you didn't use `-m`, the home directory is not created so you'll have to use the command `usermod` to change the attributes of the user or issue `mkhomedir_helper <username>` to create home directory for the user. You can also use `passwd <username>` to set/change password.

Rubric: -1 point for each independent mistake.

Q 2.2 [2 points] Login as user "`ben`" and write a command to append the line "`This line is from ben`" to `myfile.txt` file in the `cybr371`'s home directory (use absolute path). Provide the command and explain the output (i.e. did you manage to append the line? Explain why the command was successful and/or why it failed).

Solution:

```
su - ben
echo "This line is from ben" >> /home/cybr371/myfile.txt
```

It succeeds. This is because the permissions for "owner group" of the file is **rw-**, in particular, members of the owner group have "write" permission, so they can modify the file, including appending to the end of the file. The owner group of the file is `sudo`, which `ben` is a member of.

(Note that the explanation has nothing to do with the fact that the primary group is specifically "sudo". It would have worked if it had any other name.)

Rubric: 0.5 point for commands, 0.5 point for correct identification of success. 1 point for correct explanation. 2 points in total.

Q 2.3 [2 points] Login as user "`david`" now and write a command to add a line "`This is from david`" to `myfile.txt` file in `cybr371`'s home directory. (Write the command and explain why the operation is either successful or not).

Solution:

```
su - david
echo "This is from david" >> /home/cybr371/myfile.txt
```

The operation isn't successful because `david` is not the owner, or belong to the owner group, so the permissions of `other` would apply, which only have `read` permission.

Rubric: 0.5 point for commands, 0.5 point for correct identification of success. 1 point for correct explanation. 2 points in total.

Q 2.4 [2 points] Login as the user `masood` and issue a command to read the content of the file `myfile.txt` in the `cybr371`'s home directory. Can the user `masood` read the file? Write the commands and explain the output of the command.

Solution:

```
su - masood
cat /home/cybr371/myfile.txt
```

This succeeds, because `masood` is not included in the ACL, and as "other" has read permission.

Rubric: 0.5 point for commands, 0.5 point for correct identification of success. 1 point for correct explanation. 2 points in total.

Q 2.5 [3 points] after completion of step 12, Write a command to set an ACL to deny all access (read, write and execute) to `myfile.txt` for user `david`.

Solution:

```
setfacl -m u:david:--- /home/cybr371/myfile.txt
```

or equivalently:

```
setfacl -m u:david:000 /home/cybr371/myfile.txt
```

Q 2.6 [3 points] Write a command to create an ACL entry for user `mary` with write and execute permissions only on the file `myfile.txt`.

Solution:

```
setfacl -m u:mary:wx /home/cybr371/myfile.txt
```

or equivalently:

```
setfacl --modify user:mary:wx /home/cybr371/myfile.txt
```