ADS Lab 06 - Le contrôle d'accès

Authors: Vincent Peer, Pablo Urizar

Date: May the 13th, 2023

The **lab5** account on the server contains the elements asked for this lab.

Task 1: Exercises

Interpreting account and group information

> What is your UID and what is your account name?

```
$ id
uid=1007(lab6) gid=1011(lab6) groups=1011(lab6),1012(proj_a),1013(proj_b)
```

UID is 1007 and the account name is lab6.

> What is the GID of your primary group ("groupe principal") and what is its name?

```
$ id -g
1011

$ id -gn
lab6
```

GID is 1011 and its name is lab6.

> How many other groups are you a member of?

```
$ id -G | wc -w
3
```

3 groups in total, so 2 other groups from the primary.

Interpreting access control metadata on files and directories

1. For the following files, determine who is the owner, which group owns the file and characterize the group of people who can read, who can write and who can execute the file.

> /etc/passwd

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2175 Apr 27 15:28 /etc/passwd
```

- The owner `root` (rw-) has read and write permissions
- The group `root` (r--) has read permissions
- The other users (r--) have read permissions

> /bin/ls

```
$ ls -l /bin/ls
-rwxr-xr-x 1 root root 138208 Feb  7  2022 /bin/ls
```

- The owner `root` (rwx) has read, write and execute permissions
- The group `root` (r-x) has read and execute permissions
- The other users (r-x) have read and execute permissions

> ~/.bashrc

```
$ ls -l ~/.bashrc
-rw-r--r-- 1 lab6 lab6 3771 Apr 27 15:27 /home/lab6/.bashrc
```

- The owner `lab6` (rw-) has read and write permissions
- The group `lab6` (r--) has read permissions
- The other users (r--) have read permissions

> ~/.bash_history

```
$ ls -l ~/.bash_history
-rw------- 1 lab6 lab6 135 May  4 17:46 /home/lab6/.bash_history
```

- The owner `lab6` (rw-) has read and write permissions
- The group `lab6` (---) has no permissions
- The other users (---) have no permissions

2. Examine the permissions of your home directory (what option do you have to pass to ls to examine the permissions of directories?).

We can use `ls` with the `-ld` option to list details for the directory itself.

> Who is the owner and which is the owning group?

```
$ ls -ld ~
drwxr-xr-x 4 lab6 lab6 4096 May  4 15:32 /home/lab6
```

- Owner : lab6
- Owning group : lab6

> What is the configuration of permissions? Who can list files? Who can create files?

- The first character `d` indicates that it is a directory
- The owner `lab6` (rwx) has read (list), write (create), and execute permissions
- The group `lab6` (r-x) has read (list) and execute permissions
- The other users (r-x) have read (list) and execute permissions

3. What permissions allow you to create files in the /tmp directory?

As the `/tmp` directory has write permissions for others, we can create files in it:

```
$ ls -ld /tmp
drwxrwxrwt 14 root root 20480 May 11 10:15 /tmp
```

Modifying access rights

1. Using chmod in symbolic mode, create the following configurations (from initial configuration "600" == rw- --- ---):
   (With each line configured as rw- --- --- as initial configuration)
   rw- r-- --- → chmod g+r file
   rwx r-x --- → chmod u+x,g+rx file
   r-- r-- r-- → chmod a=r file
   rwx r-- r-- → chmod u+x,a+r file
   rwx --- --- → chmod u+x file

2. Conflicting permissions - Create a file (you are going to be the owner) where the permissions are configured to not allow the owner or the group to write to the file, but allow the other users to write to the file. What does the OS do if you try to write to this file?

```
$ touch test.txt
$ chmod u-w,g-w test.txt
$ chmod o+w test.txt
```

```
$ echo "test" > test.txt
-bash: test.txt: Permission denied
```

We can only read the content.

Giving other users access to your files

1. Is your colleague able to read the files in your home directory? If yes, why? If no, why not?

```
$ ls -ld /home/lab5
drwxr-xr-x 7 lab5 lab5 4096 May  9 14:41 /home/lab5
```

From the user `lab6` (other), we have read and execute permissions to the home directory of `lab5`. So yes, we can read `lab5` files from `lab6` user.

2. What do you need to do so that your colleague (and maybe others) can read your files? What do you need to do so that nobody else can read your files?

Everybody can already read files so we have nothing more to do.
To restrict permissions, we could create a group and include in it every person that we would like to have read permissions to our home directory and exlude every other user.

```
$ sudo groupadd lab6_read
```

Then add every user that we want to have access to our home directory with :

```
sudo usermod -aG lab6_read user1
```

Finally we remove permissions to others :

```
$ chmod -R o-rwx /home/lab6
```

If we want that nobody else can read our files, we can directly use this last command without creating a group.

3. In your home directory create a directory named shared . By using the commands chmod and chgrp configure the directory in such a way that only your colleague is able to read the directory and its files. You can use the groups proj_a and proj_b . Both you and your colleague are already a member of these groups. (For the sake of this exercise, suppose that nobody else is member of this group, only you and your colleague.) Give your colleague also access rights to create new files and modify existing files. What commands did you use?

Create the directory :

```
$ mkdir shared
```

Set the group owner of the directory created before to `proj_a` :

```
$ chgrp proj_a shared
```

Remove permissions for others :

```
$ chmod o-rwx shared
```

Grant write permission to the members of the group :

```
$ chmod g+w shared
```

Find

1. What does find do with hidden files or directories?

All hidden files and directories are displayed with a recursive display for directories.

2. Using find display all the files in your home directory

- that end in .c , .cpp or in .sh

```
$ find ~ -type f \( -name "*.c" -o -name "*.cpp" -o -name "*.sh" \)
/home/lab6/.zsh/gitstatus.sh
/home/lab6/.zsh/theme-simple.sh
/home/lab6/.zsh/theme-full.sh
/home/lab6/.zsh/title.sh
/home/lab6/.zsh/theme.sh
/home/lab6/.zsh/history.sh
/home/lab6/.zsh/generic.sh
/home/lab6/.zsh/keys.sh
/home/lab6/.zsh/alias.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/gitstatus.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/theme-simple.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/theme-full.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/title.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/theme.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/history.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/generic.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/keys.sh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/completion.sh
```

- that are executable

```
$ find ~ -type f -executable
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/zsh-syntax-highlighting/highlighters/main/main-highlighter.zsh
/home/lab6/.zsh/_tmp_states/arcanite_gestion/users/features/config_files/Zshconfig
/.zsh/scripts/gitstatus.py
```

- that have not been modified since more than two years

```
$ find ~ -type f -mtime +730
```

- that have not been accessed since more than two years

```
$ find ~ -type f -atime +730
```

- that have not been accessed since more than three years and that are bigger than 3 MB (good candidates for cleanup)

```
$ find ~ -type f -atime +1095 -size +3M
```

Display all the directories in your home directory that are called .git (probably the root of a git repository)

```
$ find ~ -type d -name ".git"
```

3. Suppose your current directory has no subdirectories. You want to display all files that contain the word root . Which of the two commands is correct and why:

```
find . -type f -exec grep -l 'root' {} \;
find * -type f -exec grep -l 'root' {} \;
```

The first command indicates to *find* that it has to use the current directory to look for files that contains the word root. With the *, the shell uses globbing to give every file in the current directory to the find function. The first command using . is probably better because we use entirely the find functionalities and no other as the globbing. Furthermore, hidden files are displayed with this first command, it can be usefull.

Script task 2-5:

```

```

```
#!/bin/bash
#
# Description : Display world-writable files for a specific directory. A world-
writable file is a file with the write bit set for others.
# It additionally suggest to the user to fix this risk by removing this access for
others. Finally, it displays the group writable files/directories.
#
# Authors : Pablo Urizar, Vincent Peer
#
# Date : 14.05.2023
```

```
if [[ $# -ne 1 ]]
then
        echo "Error: missing argument. Please specify a directory" >&2
        exit  1

elif [[ ! -d "$1" ]]
then
        echo "Invalid directory" >&2
        exit 1

else
        echo "The following files/directories are world-writable:"
        find "$1" -perm -o+w
fi

read -p "Do you want the permissions to be fixed (y/n)?" response

if [[ "${response}" == "y" || "${response}" == "yes" ]]
then
        find "$1" -perm -o+w -exec chmod o-w {} \;
        echo "Offending permissions have been removed"
else
        echo "Offending permissions unchanged"
fi

echo "The following files/directories are writable for groups:"
find "$1" ! -group $USER -perm -g+w
```

Listing of the test_dir directory

```
test_dir:
total 28
drwxrwxrwx 3 lab5 lab5   4096 May  4 16:43 a
drwxrwxr-x 3 lab5 lab5   4096 May  4 16:29 f
-rwxrwxrwx 1 lab5 lab5     37 May  4 16:28 info.txt
drwxrwxr-x 2 lab5 proj_a 4096 May  9 14:45 proj_a
drwxrwxr-x 2 lab5 lab5   4096 May  9 14:30 proj_a2
```

```
drwxrwxr-x 2 lab5 proj_b 4096 May  9 14:23 proj_b
drwxr-xr-x 2 lab5 lab5   4096 May  9 14:31 proj_b2

test_dir/a:
total 12
drwxrwxr-x 2 lab5 lab5 4096 May  4 16:44 b
-rwxrwxrwx 1 lab5 lab5   31 May  4 16:28 file1
-rw-rw-rw- 1 lab5 lab5   22 May  4 16:43 file2

test_dir/a/b:
total 8
-rw-rw-r-- 1 lab5 lab5 32 May  4 16:28 fileb
-rw-rw-rw- 1 lab5 lab5 39 May  4 16:44 unprotectedFile

test_dir/f:
total 8
-rw-rw-r-- 1 lab5 lab5   25 May  4 16:29 file
drwxrwxrwx 2 lab5 lab5 4096 May  4 16:30 g

test_dir/f/g:
total 4
-rw-rw-rw- 1 lab5 lab5 28 May  4 16:30 file.txt

test_dir/proj_a:
total 0
-rw-rw-r-- 1 lab5 proj_a 0 May  9 14:45 README

test_dir/proj_a2:
total 0

test_dir/proj_b:
total 0

test_dir/proj_b2:
total 0
```