Sicurezza e Privatezza

LAB 3 – Permissions and access control

The learning objective of this lab is for students to get the basics of file security through file ownership, file permissions and access control. After finishing the lab, students should be able to understand the basic of Linux access control mechanism.

Students are invited to complete the following exercises by write the command for solve each question in a txt file and submit it to upload.di.unimi.it. The name of the txt file should be `Lab3NameSurnameUniversityid` (eg. Lab3MatteoZoia856378.txt). Files not conforming to these rules will be automatically rejected.

The first ten students who will respond correctly to the entire lab will receive a bonus of 0.5pt on the final grade.

Exercise #1: Permissions

1. As normal user create a directory `~/permissions`. Create a file owned by yourself in there. Copy a file owned by root from `/etc/` to your permissions dir, who owns this file now?
2. As root, create a file in the users `~/permissions` directory. As normal user, look at who owns this file created by root?
3. Change the ownership of all files in `~/permissions` to yourself.
4. Make sure you have all rights to these files, and others can only read.
5. With *chmod*, is 770 the same as `rwxrwx---` ?
6. With *chmod*, is 664 the same as `r-xr-xr--` ?
7. With *chmod*, is 400 the same as `r--------` ?
8. With *chmod*, is 734 the same as `rwxr-xr--` ?
9. Display the *umask* in octal and in symbolic form.
10. Set the *umask* to 077, but use the symbolic format to set it. Verify that this works.
11. Create a file as root, give only read to others. Can a normal user read this file? Test writing to this file with *vi/vim/nano/emacs* (your favorite editor).
12. Create a file as normal user, give only read to others. Can another normal user read this file? Test writing to this file with an editor.
13. Can root read this file? Can root write to this file with an editor?
14. Create a directory that belongs to a group, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

Exercise #2: Sticky bit, GID and UID

1. Set up a directory, owned by the group students.
2. Members of the students group should be able to create files in this directory.
3. All files created in this directory should be group-owned by the students group.
4. Users should be able to delete only their own user-owned files.
5. Verify the permissions on `/usr/bin/passwd`. Remove the *setuid*, then try changing your password as a normal user. Reset the permissions back and try again.

Exercise #3: Users and groups

1. Show all groups that your current user is in.
2. Add new group to your machine named *school,* create a new user called George and add it to the group school.
3. Create a file named `homework.txt` with your current user. Print the **acls** (with the binary `/usr/bin/getfacl`) associated with your actual user and the file.
4. Add user George with octal permission `7` to the acl of file `homework.txt` (use the binary `setfacl`). Then add the group school with octal permission `6` to the acl of the same file. Show the results with `getfacl.`
5. With the setfacl binary remove an acl entry for George from the `homework.txt` file. Can you still write the file?

Exercise #4: Links

1. Create two files named `poem.txt` and `tale.txt`, put some text in them.
2. Create a hard link to `poem.txt` named `hlpoem.txt`.
3. Display the inode numbers of these three files, the hard links should have the same inode.
4. Use the find command to list the two hardlinked files
5. Everything about a file is in the inode, except two things : name them.
6. Create a symbolic link to `tale.txt` called `sltale.txt`.
7. Find all files with inode number 2. What does this information tell you ?