File Permissions in Linux

Project Description

In this project, I need to use Linux commands to configure authorization.

In Linux, file and directory permissions specify who can access specific files and directories. So I will explore files and directories to change the ownership and limit who has access.

Here is the scenario:
I will examine and manage the permissions on the files and directories in the **/home/researcher2/projects** directory for the **researcher2** user, who is part of the **research_team** group

I'll check the permissions of all the files in the directory, even the hidden files, and make sure that they align with the authorization given - if it doesn't I'll change the permissions.

Check File and Directory Details

```
researcher2@f619fcd20110:~$ ls
projects
researcher2@f619fcd20110:~$ cd projects
researcher2@f619fcd20110:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
researcher2@f619fcd20110:~/projects$
```

Here I am exploring the permissions of the **projects** directory and the files it contains. Using the **ls -l** command, we can see that user **reseacher2** has permission to access these 20 files and directories, as well as how the permissions are set via the directories on the left-hand side. Group **research_team** owns all of these files.

```
researcher2@f619fcd20110:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 02:43 .
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 03:30 ..
-rw--w---- 1 researcher2 research_team   46 Feb  9 02:43 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
researcher2@f619fcd20110:~/projects$
```

If we use the command **ls -la**, we can now see the hidden files within the project's directory. Specifically, the **.projects_x.txt** file.

Describe the Permissions String

```
drwxr-xr-x
drwxr-xr-x
-rw--w----
drwx--x---
-rw-rw-rw-
-rw-r-----
-rw-rw-r--
-rw-rw-r--
```

If we look back at the last command, we can see how the permissions are set via the directories on the left-hand side of the files. There is a 10-character string that indicates how permissions on a file are set in Linux with full permissions being: **drwxrwxrwx**

To better understand this, let's go over the read, write, and execute permissions:

- **read**: this is the ability to read the file contents; for directories, this is the ability to read all contents in the directory including both files and subdirectories.
- **write**: this is the ability to make modifications to the file contents; for directories, this is the ability to create new files in the directory.
- **execute**: this is the ability to execute the file if it's a program; for directories, this is the ability to enter the directory and access its files.

Another thing to go over are the three owner types:

- **user**: the owner of the file
- **group**: a larger group that the owner is a part of
- **other**: all other users on the system

Now that we understand permissions and owner types, let's break down this 10-character string further to understand how permissions work:

- The 1st character indicates the file type - **d** indicates a directory.
- The 2nd - 4th characters indicate the read (r), write (w), and execute (x) permissions for the **user.**

- The 5th-7th characters indicate the read (r), write (w), and execute (x) permissions for the **group.**
- The 8th-10th characters indicate the read (r), write (w), and execute (x) permissions for the owner type of **'other'.**
- **Whenever you see a character that has a hyphen (-), that indicates that permission is not granted within the user, group, or other groups.**

```
researcher2@dd5ac411cdb8:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:33 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Feb  9 02:33 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  9 02:33 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_t.txt
```

For example, if we look at **project_r.txt** we can see that**:**

- User have read and write permissions
- Group have read and write permissions
- Other have only read permissions

Change File Permissions

```
researcher2@f619fcd20110:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
```

What if we want to change the file permissions? For example, **project_k.txt** currently has write permissions for the other user group. If we would like to remove these permissions we would use the **chmod** command:

```
chmod o-w project_k.txt
```

Let's break this command down:

- **chmod** - helps change permissions on files and directories.
- **o** - **u** sets the permission for the *user*, **g** sets the permission for the *group*, and in this case; **o** sets the permission for the *'other'* group
- **-w** - a remove (-) or an add (+) indicates whether we are adding or removing a permission, while (w) indicates the write permission. So, in this case, we are removing the write permission.
- **Project_k.txt** - this is the file we are setting the change in permission to

```
researcher2@f619fcd20110:~/projects$ chmod o-w project_k.txt
researcher2@f619fcd20110:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
researcher2@f619fcd20110:~/projects$
```

Change File Permissions on a Hidden File

If we go back to our hidden files, we can see that **.project_x.txt** has read and write permissions for the user owner as well as write permissions for the group owner:

```
researcher2@f619fcd20110:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 02:43 .
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 03:30 ..
-rw--w---- 1 researcher2 research_team   46 Feb  9 02:43 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw------- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
researcher2@f619fcd20110:~/projects$
```

Let's say we want to remove all write permissions entirely on this archived file. To do this, we'll need to use the chmod command again:

```
chmod u-w,g-w,g+r .project_x.txt
```

Let's break this command down:

- **chmod** - helps change permissions on files and directories.
- **u-w** - **u** sets the permission for the user, while **-w** means we will remove write permission from the user owner.
- **g-w** - **g** sets the permissions for the group, while **-w** means we will remove write permission from the group owner.
- **g+r** - **g** sets the permissions for the group, while **+r** means we will add read permission to the group owner.
- **.Project_x.txt** - this is the file we are setting the change in permission to

```
researcher2@f619fcd20110:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@f619fcd20110:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 02:43 .
drwxr-xr-x 3 researcher2 research_team 4096 Feb  9 03:30 ..
-r--r----- 1 researcher2 research_team   46 Feb  9 02:43 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:43 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_k.txt
-rw------- 1 researcher2 research_team   46 Feb  9 02:43 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:43 project_t.txt
researcher2@f619fcd20110:~/projects$
```

As you can see, the execute permission has been removed from the **drafts** directory.

Change Directory Permissions

```
researcher2@dd5ac411cdb8:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Feb  9 02:33 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_k.txt
-rw------- 1 researcher2 research_team   46 Feb  9 02:33 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_t.txt
researcher2@dd5ac411cdb8:~/projects$
```

Currently, we have the **drafts** directory within our **projects** directory. If we want to change the directory permissions of **drafts**, we need to confirm if we have execute permissions to access the drafts directory.
(looking to our left-hand side, we can see that We do have an execute permission (x) in the user owner. So yes, the group has execute permissions and therefore has access to the **drafts** directory.

To remove the execute permissions for the group from the **drafts** directory, we'll use the chmod command one last time:

```
chmod g-x drafts
```

Let's break this command down:

- **chmod** - helps change permissions on files and directories.
- **g-x** - **g** sets the permissions for the group, while **-x** means we will remove the execute permission from the group owner.
- **drafts** - this is the directory we are setting the change in permission to

```
researcher2@dd5ac411cdb8:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Feb  9 02:33 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_k.txt
-rw------- 1 researcher2 research_team   46 Feb  9 02:33 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  9 02:33 project_t.txt
researcher2@dd5ac411cdb8:~/projects$
```

As you can see, the execute permission has been removed from the **drafts** directory (the x is gone on the left-hand side).

Summary

In summary, we used the Linux Bash shell commands to configure authorization by managing files and directory access. We went over the 10-character permission string, the three owner types, and the **chmod** command for changing file/directory permissions. With this knowledge, we are better equipped to handle Linux as a cybersecurity analyst.