File permissions in Linux

Project description

To ensure the security of the project's directory, the research team at my organization required the file permissions for specific files and directories to be updated. The current permissions did not align with the necessary authorization level. I undertook the following tasks to review and adjust the permissions:

Check file and directory details

Within this code, I utilized Linux commands in order to determine the permissions that were previously established on a particular folder within the file structure.

```
researcher2@f1d01084e048:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jun 15 23:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Jun 15 23:39 ..
-rw--w---- 1 researcher2 research_team   46 Jun 15 23:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jun 15 23:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jun 15 23:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jun 15 23:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 15 23:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 15 23:08 project_t.txt
researcher2@f1d01084e048:~/projects$
```

The first line of the screenshot displays the command I entered, and the other lines display the output. The code lists all contents of the projects directory. I used the **ls** command with the **-la** option to display a detailed listing of the file contents that also returned hidden files. The output of my command indicates that there is one directory named drafts, one hidden file named **.project_x.txt**, and five other project files. The 10-character string in the first column represents the permissions set on each file or directory.

Describe the permissions string

The 10-character string can be analyzed to identify the individuals who are authorized to access the file and the specific permissions they have.. The characters and what they represent are as follows:

● 1st character: This character is either a **d** or **hyphen (-)** and indicates the file type. If it's a **d**, it's a directory. If it's a **hyphen (-)**, it's a regular file.

● 2nd-4th characters: These characters indicate the **read (r), write (w)**, and **execute (x)** permissions for the user. When one of these characters is a **hyphen (-)** instead, it indicates that this permission is not granted to the user.
● 5th-7th characters: These characters indicate the **read (r), write (w)**, and **execute (x)** permissions for the group. When one of these characters is a **hyphen (-)** instead, it indicates that this permission is not granted for the group.
● 8th-10th characters: These characters indicate the **read (r), write (w)**, and **execute (x)** permissions for other. This owner type consists of all other users on the system apart from the user and the group. When one of these characters is a **hyphen (-)** instead, that indicates that this permission is not granted for other.
For example, the file permissions for **project_t.txt** are **-rw-rw-r--.** Since the first character is a **hyphen (-)**, this indicates that **project_t.txt is** a file, not a directory. The second, fifth, and eighth characters are all **r**, which indicates that user, group, and other all have read permissions. The third and sixth characters are **w**, which indicates that only the user and group have write permissions. No one has **execute** permissions for **project_t.txt.**

Change file permissions

Following the organization's decision to restrict write access to files for **other**, I reviewed the file permissions I had previously obtained. Based on my analysis, I found that the write access for **other** needed to be removed from **'project_k.txt'** to comply with this requirement.

```
researcher2@f1d01084e048:~/projects$ chmod o-w project_k.txt
researcher2@f1d01084e048:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jun 15 23:08 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jun 15 23:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jun 15 23:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 15 23:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 15 23:08 project_t.txt
researcher2@f1d01084e048:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. The chmod command changes the permissions on files and directories. The first argument indicates what permissions should be changed, and the second argument specifies the file or directory. In this example, I removed write

permissions from other for the **project_k.txt** file. After this, I used **ls -l** to review the updates I made.

Change file permissions on a hidden file

The research team at my organization recently archived project_x.txt. They do not want anyone to have write access to this project, but the user and group should have read access. The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@ccec5b444195:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@ccec5b444195:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jun 16 00:07 .
drwxr-xr-x 3 researcher2 research_team 4096 Jun 16 00:12 ..
-r--r----- 1 researcher2 research_team   46 Jun 16 00:07 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jun 16 00:07 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_k.txt
-rw------- 1 researcher2 research_team   46 Jun 16 00:07 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_t.txt
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I know **.project_x.txt** is a hidden file because it starts with a period (.). In this example, I removed write permissions from the user and group, and added read permissions to the group. I removed write permissions from the user with **u-w**. Then, I removed write permissions from the group with **g-w**, and added read permissions to the group with **g+r.**

Change directory permissions

In my organization, only the researcher2 user should be granted access to the drafts directory and its contents. This means that no one other than researcher2 should have execute permissions.

The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@ccec5b444195:~/projects$ chmod g-x drafts
researcher2@ccec5b444195:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jun 16 00:07 .
drwxr-xr-x 3 researcher2 research_team 4096 Jun 16 00:12 ..
-r--r----- 1 researcher2 research_team   46 Jun 16 00:07 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Jun 16 00:07 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_k.txt
-rw------- 1 researcher2 research_team   46 Jun 16 00:07 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun 16 00:07 project_t.txt
```

The output here displays the permission listing for several files and directories. Line 1 indicates the current directory **(projects),** and line 2 indicates the parent directory **(home)**. Line 3 indicates a regular file titled **.project_x.txt**. Line 4 is the directory **(drafts)** with restricted

permissions. Here you can see that only researcher2 has execute permissions. It was previously determined that the group had execute permissions, so I used the chmod command to remove them. The researcher2 user already had execute permissions, so they did not need to be added.

Summary

To align with my organization's desired level of authorization, I modified numerous permissions for files and directories within the projects directory. As a first step, I used **ls -la** to review directory permissions. This information guided my subsequent actions. I then used the **chmod** command multiple times to adjust file and directory permissions.