File permissions in Linux

Project description

The research team at my organization must adjust file permissions for specific files and subdirectories within the **projects** directory. At present, the assigned permissions do not align with the appropriate level of authorization. Reviewing and correcting these settings is necessary to maintain system security. To carry out this task, I took the following steps:

Check file and directory details

The code below shows how I used Linux commands to identify the current permissions applied to a particular directory in the file system.

```
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-rw--w---- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

At the top of the screenshot, you can see the command I entered, followed by the system's response. This command generates a complete listing of the **projects** directory. To accomplish this, I used `ls -la`, which produces a long-format display that not only shows file details but also includes hidden files that would normally be omitted.

From the output, it's clear that the directory contains a mix of items: a subdirectory named **drafts**, a hidden file titled **.project_x.txt**, and five other project-related files. Each entry in the listing begins with a 10-character string. This string encodes the file type along with the access permissions granted to different categories of users, making it a crucial part of interpreting the results.

Describe the permissions string

This 10-character sequence reveals both the type of item (file or directory) and the permissions assigned to the user, the group, and all others. The breakdown works as follows:

- **First character** – identifies the item type. A **d** means directory, while a hyphen (**-**) means it is a regular file.

- **Characters 2–4** – show the user's permissions: read (**r**), write (**w**), and execute (**x**). A hyphen in place of one of these letters indicates that the permission is not granted.

- **Characters 5–7** – show the group's permissions in the same way (read, write, execute). Again, a hyphen means the permission is missing.

- **Characters 8–10** – display permissions for "others," meaning all users outside of the file's owner and group. As with the others, a hyphen indicates the absence of a specific permission.

For instance, the permissions string for **project_t.txt** is `-rw-rw-r--`. The initial hyphen indicates it is a file rather than a directory. Both the user and the group have read (**r**) and write (**w**) permissions, while all other users have only read (**r**) access. No one has execute (**x**) permissions for this file.

Change file permissions

The organization decided that the "other" user group should not have write access to any files. To address this requirement, I reviewed the permission settings retrieved earlier and identified that **project_k.txt** allowed write access for others. This permission needed to be revoked.

The command below shows how I used Linux tools to remove that access.

```
researcher2@5d738f0f927b:~/projects$ chmod o-w project_k.txt
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-rw--w---- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

In the screenshot, the first two lines show the commands I entered, while the remaining lines display the system's response to the second command. The `chmod` utility is used to modify access rights for files and directories. Its first argument specifies the permission changes to apply, and the second argument identifies the target file or directory. In this case, I removed write access for "others" on the file **project_k.txt**. To verify the change, I ran `ls -la` again, which confirmed that the permissions had been updated successfully.

Change file permissions on a hidden file

The research team recently archived **project_x.txt**, a hidden file within the system. To preserve its integrity, they requested that no one be able to write to the file, while still allowing both the owner and the group to retain read access.

The code below demonstrates how I applied the necessary permission changes using Linux commands.

```
researcher2@3213bbc1d047:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@3213bbc1d047:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 ..
-r--r----- 1 researcher2 research_team   46 Dec 20 15:36 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 20 15:36 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec 20 15:36 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 20 15:36 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 20 15:36 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 20 15:36 project_t.txt
researcher2@3213bbc1d047:~/projects$
```

At the top of the screenshot are the commands I executed, followed by the output from the second command. The file **.project_x.txt** is identified as hidden because its name begins with a period (**.**). For this task, I modified its permissions so that both the user and the group no longer have write access, while the group was also granted read access. These changes were made using the following adjustments: `u-w` removed write privileges from the user, `g-w` removed write privileges from the group, and `g+r` added read privileges for the group.

Change directory permissions

The organization required that access to the **drafts** directory be restricted solely to the user **researcher2**. To enforce this policy, execute permissions had to be removed for all other users, ensuring that no one besides researcher2 could navigate into or work with the directory's contents.

The commands below show how I applied the necessary permission changes in Linux.

```
researcher2@5d738f0f927b:~/projects$ chmod g-x drafts
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-r--r----- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

The output shown in the screenshot lists permission details for several files and directories. The first line represents the **projects** directory itself, while the second line points to its parent directory (**home**). The third entry is a hidden file named **.project_x.txt**, and the fourth entry is the **drafts** directory, which has restricted access. In this case, only the **researcher2** user retains execute permissions. Previously, the group also had execute rights, but I removed them using the `chmod` command. Since **researcher2** already had the appropriate permissions, no additional changes were required for that account.

Summary

I modified file and directory permissions within the **projects** directory to align with

organizational security requirements. I began by running `ls -la` to examine the existing permission structure. Based on that information, I applied the `chmod` command to adjust access rights where necessary. The primary goal was to ensure that only **researcher2** had execute access to the **drafts** directory, preventing other users or groups from entering or modifying its contents. The code below shows the Linux commands used to make these changes.