File permissions in Linux

Project description

The research team at my organization needed to revise the file permissions for specific files and directories within the **projects** directory. The existing permissions did not accurately reflect the appropriate levels of access required. To enhance system security and ensure proper authorization, I carried out a review and updated the necessary permissions accordingly.

Check file and directory details

```
researcher2@1f3d02bd0ca2:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Apr  8 08:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Apr  8 08:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  8 08:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_t.txt
researcher2@1f3d02bd0ca2:~/projects$
```

The first line in the screenshot shows the command I executed, while the remaining lines show the corresponding output. I used the `ls -la` command to generate a detailed listing of all items within the *projects* directory, including hidden files. Based on the output, the directory contains a subdirectory called **drafts**, a hidden file named **.project_x.txt**, and five additional project files. Each item's permissions are shown in the first column as a 10-character string, indicating the current access settings for that file or folder.

Describe the permissions string

The 10-character permission string can be broken down to reveal who has access to a file or directory, along with the specific level of access granted. Here's how each part of the string works:

- **Character 1**: Indicates the type of item. A **d** means it's a directory, while a **-** denotes a regular file.
- **Characters 2–4**: Show the permissions assigned to the file owner. These include read (**r**), write (**w**), and execute (**x**) permissions. If any of these are replaced by a **-**, that specific permission is not granted.
- **Characters 5–7**: Reflect the permissions assigned to the group associated with the file, using the same **r**, **w**, and **x** indicators—or a **-** if the permission is absent.

- **Characters 8–10**: Represent the permissions for all other users (those who are not the owner or part of the group). These follow the same format: **r**, **w**, **x**, or **-**.

Take, for example, the file **project_t.txt**, which has permissions set to **-rw-rw-r--**. The initial **-** confirms that it's a standard file rather than a folder. Both the user and the group have read and write privileges (as shown by the **rw** in their respective sections), while everyone else has read-only access. No execute rights are granted to any user type.

Change file permissions

The organization has established that no files should allow write access for users outside of the owner and group. With this policy in mind, I reviewed the file permissions obtained earlier and identified that **project_k.txt** needed its write access for "other" users revoked.

The first two lines reflect the commands I entered, while the following lines display the result of the second command. I used the `chmod` utility, which is designed to modify permissions on files and directories. The first parameter defines which permissions are being altered, and the second specifies the target file or folder.

```
researcher2@1f3d02bd0ca2:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  8 08:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  8 08:20 ..
-rw--w---- 1 researcher2 research_team   46 Apr  8 08:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Apr  8 08:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Apr  8 08:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  8 08:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_t.txt
researcher2@1f3d02bd0ca2:~/projects$
```

In this case, I removed write access from "other" users for **project_k.txt**. After applying the change, I ran `ls -la` to verify that the updated permissions had been applied successfully.

Change file permissions on a hidden file

In the screenshot, the first two lines show the commands I typed, while the rest of the output comes from the second command. The file **.project_x.txt** is identified as hidden due to the leading dot (.), which is a common convention in Unix-based systems.

In this scenario, I adjusted the permissions by removing write access from both the user and the group. Additionally, I granted read access to the group. These changes were made using the `chmod` command with the following options: `u-w` to revoke write permissions from the user, `g-w` to do the same for the group, and `g+r` to provide the group with read access.

```
researcher2@1f3d02bd0ca2:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@1f3d02bd0ca2:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  8 08:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  8 08:20 ..
-r--r----- 1 researcher2 research_team   46 Apr  8 08:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Apr  8 08:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Apr  8 08:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  8 08:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_t.txt
researcher2@1f3d02bd0ca2:~/projects$
```

Change directory permissions

```
researcher2@1f3d02bd0ca2:~/projects$ chmod g-x drafts
researcher2@1f3d02bd0ca2:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Apr  8 08:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Apr  8 08:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  8 08:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  8 08:08 project_t.txt
researcher2@1f3d02bd0ca2:~/projects$
```

The initial two lines in the screenshot show the commands I executed, while the rest displays the output from the second command. Based on earlier findings, I identified that the group had execute permissions on the file, which needed to be removed. Using the `chmod` command, I successfully revoked those execute rights for the group. Since the user **researcher2** already had execute permissions, no additional changes were necessary for that user.

Summary

I made several permission changes to ensure that files and directories within the **projects** directory aligned with my organization's access control policies. The process began by running the `ls -la` command, which provided a comprehensive list of all items in the directory—along with their associated permissions, ownership, and visibility (including hidden files).

Reviewing this detailed output helped me identify which files and directories required updates based on our security guidelines. Using this information, I applied the `chmod` command multiple times to adjust permissions accordingly. These changes included actions such as removing write or execute access from unauthorized users or groups, granting read access where appropriate, and ensuring that sensitive files were not exposed to unintended users.