Demonstrating File Permission Management in Linux

Project description

The research team required updates to the file and directory permissions within the `/home/researcher2/projects` directory to align with the organization's authorization standards. To address this, I followed a series of steps to secure files and directories while ensuring proper access for the `researcher2` user, a member of the `research_team` group.

To begin, I navigated to the `projects` directory using the `cd` command and used `ls -l` and `ls -la` to review the permissions for all files and directories, including hidden files. This review highlighted several discrepancies in access rights that required correction.

I performed the following actions to update the permissions:

- **Secured `project_k.txt`:** Removed write permissions for others using `chmod o-w project_k.txt` to prevent unauthorized modifications.
- **Updated `project_m.txt`:** Revoked read permissions for the group with `chmod g-r project_m.txt` to limit exposure of sensitive data.
- **Configured `.project_x.txt`:** Adjusted permissions on the hidden file using `chmod u-w,g-w,g+r .project_x.txt`, ensuring the group retained read access while removing write permissions for both the user and group.
- **Restricted the `drafts` directory:** Removed execute permissions for the group using `chmod g-x drafts` to prevent unauthorized access to its contents.

These changes ensured that permissions were correctly configured to meet the required authorization levels. By restricting unauthorized access and protecting sensitive data, the system's security posture was strengthened.

Check file and directory details

To analyse the existing permissions within the `/home/researcher2/projects` directory, I began by navigating to the directory using the command `cd projects`. Once inside, I needed to view the details of all files, including their permissions, ownership, and other metadata.

To accomplish this, I executed the `ls -l` command, which provided a detailed list of the directory's contents. This output included information about each file and directory, such as the 10-character string representing the current permissions.

```
researcher2@94c735324176:~$ cd projects
researcher2@94c735324176:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

To ensure no hidden files were overlooked, I followed up with the `ls -la` command. This command displayed all contents of the directory, including hidden files starting with a period.

```
researcher2@94c735324176:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 21:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 22:30 ..
-rw--w---- 1 researcher2 research_team   46 Oct 11 21:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

The output revealed one directory named `drafts`, one hidden file named `.project_x.txt`, and several other project files. By combining these commands, I obtained a comprehensive overview of all files and their current permissions, allowing me to identify and address any discrepancies.

Describe the permissions string

The 10-character string provides detailed information about who can access a file or directory and the specific permissions assigned. Using the example `drwxr-xr-x` based on the first line in the image above, here's how the characters are interpreted:

- **1st character**: Identifies the type of file. A `d` indicates it is a directory, while a hyphen (`-`) represents a regular file.

- **2nd-4th characters**: Show the read (`r`), write (`w`), and execute (`x`) permissions for the **user** (file owner). A hyphen (`-`) in any position means that specific permission is not granted to the user.
- **5th-7th characters**: Represent the read (`r`), write (`w`), and execute (`x`) permissions for the **group**. A hyphen (`-`) in any position indicates that the corresponding permission is not granted to the group.
- **8th-10th characters**: Indicate the read (`r`), write (`w`), and execute (`x`) permissions for **others** (users outside the owner and group). A hyphen (`-`) in these positions means the permission is not granted to others.

Additional example:

The file permissions for `project_t.txt` are `-rw-rw-r--`. Here's what they indicate:

- The first character is a hyphen (`-`), meaning it is a regular file, not a directory.
- The second, fifth, and eighth characters are `r`, indicating that the **user**, **group**, and **others** all have **read** permissions.
- The third and sixth characters are `w`, meaning that only the **user** and **group** have **write** permissions.
- No `x` appears, showing that no one has **execute** permissions for `project_t.txt`.

Change file permissions

The organization required that the `other` category should not have write access to any files in the `projects` directory. Upon reviewing the permissions of all files using the `ls -la` command, I identified that the file `project_k.txt` allowed write permissions for others, which posed a security risk.

```
researcher2@94c735324176:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 21:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 22:30 ..
-rw--w---- 1 researcher2 research_team   46 Oct 11 21:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

To resolve this, I used the `chmod` command to remove write permissions from `other` for the file `project_k.txt`. The first argument of the command specifies the change to the permissions, while the second argument identifies the file being updated. In this case, I executed the command chmod o-w project_k.txt. Afterward, I verified the updated permissions using `ls -l` to confirm the change.

```
researcher2@94c735324176:~/projects$ chmod o-w project_k.txt
researcher2@94c735324176:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
```

Similarly, I addressed the permissions for the restricted file `project_m.txt`. It was determined that the group should not have read access to this file. Using the output from `ls -l`, I confirmed that the group had read permissions. To comply with security requirements, I executed the following command chmod g-r project_m.txt. This adjustment ensured that only the file's owner could access the contents of `project_m.txt`, restricting access from both the group and others. Finally, I reviewed the updated permissions using `ls -la` to ensure that all changes aligned with organizational security policies.

```
researcher2@94c735324176:~/projects$ chmod g-r project_m.txt
researcher2@94c735324176:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 21:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 22:30 ..
-rw--w---- 1 researcher2 research_team   46 Oct 11 21:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

Change file permissions on a hidden file

The research team recently archived `.project_x.txt` and decided that no one should have write access to this file, while both the user and group should retain read access. To implement this, I used the following Linux commands to adjust the file's permissions:

```
researcher2@94c735324176:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@94c735324176:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 21:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 22:30 ..
-r--r----- 1 researcher2 research_team   46 Oct 11 21:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

The first two lines of the screenshot show the commands I entered, and the following lines display the output after executing the second command. I identified that `.project_x.txt` is a hidden file because it starts with a period (.).

In this case, I removed write permissions from both the user and the group. To do this, I used the `chmod` command with `u-w` to remove write permissions from the user and `g-w` to remove write permissions from the group. Additionally, I added read permissions to the group using `g+r`. This ensured that the user and group could read the file, but neither could modify it.

Finally, I verified the changes by checking the updated permissions using `ls -la`.

Change directory permissions

```
researcher2@94c735324176:~/projects$ chmod g-x drafts
researcher2@94c735324176:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 21:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 11 22:30 ..
-r--r----- 1 researcher2 research_team   46 Oct 11 21:15 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Oct 11 21:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 11 21:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 11 21:15 project_t.txt
researcher2@94c735324176:~/projects$
```

The drafts directory required permission adjustments to ensure that only the `researcher2` user could access it. Using the `ls -l` command, I reviewed the current permissions, which showed that the group had execute permissions. To restrict access, I used the `chmod g-x drafts` command to remove the execute permissions for the group. The `researcher2` user already had the necessary execute permissions, so no further changes were needed. This adjustment secured the drafts directory against unauthorized group access.

Summary

I managed and configured file and directory permissions within the `/home/researcher2/projects` directory to align with the required level of authorization. Using the `ls` and `ls -la` commands, I reviewed the existing permissions, which informed my subsequent actions. I then used the `chmod` command to modify permissions, ensuring sensitive files and directories were secured against unauthorized access or modification.