File permissions in Linux

Project description

As part of internal security operations, I conducted a permissions audit and configuration task on a Linux-based research server. The goal was to ensure that sensitive project files were appropriately restricted and that unauthorized write access was removed according to company policy. This involved checking and modifying file and directory permissions using Linux commands to enforce least privilege access across team resources.

Check file and directory details

`ls -la`

```
researcher2@794a61d0f2ab:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 26 08:49 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 26 08:50 ..
-rw--w---- 1 researcher2 research_team   46 Jul 26 08:49 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 26 08:49 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jul 26 08:49 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 26 08:49 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 26 08:49 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 26 08:49 project_t.txt
researcher2@794a61d0f2ab:~/projects$
```

Explanation:

- This command lists all files in the `projects` directory, including hidden files (those beginning with a `.`), along with their detailed permission strings, ownership, and timestamps.

Describe the permissions string

Selected file: `.project_x.txt`
Permissions string: `-rw-r-----`

Explanation: This 10-character string represents the file type and permission settings for the file `.project_x.txt`.

- The first character `-` indicates that this is a **regular file**.
- The next three characters `rw-` indicate that the **owner** (`researcher2`) has **read (r)** and **write (w)** permissions, but **no execute (-)** permission.
- The next three characters `r--` show that the **group** (`research_team`) has **read-only** access.
- The last three characters `---` indicate that **others** have **no permissions** for this file.

This string ensures that only the owner can view and edit the file, while group members can only view it, and all others are restricted.

Change file permissions

```
chmod g-w,o-w project_t.txt
```

```
researcher2@794a61d0f2ab:~/projects$ chmod g-w,o-w project_t.txt
researcher2@794a61d0f2ab:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 26 08:49 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 26 08:50 ..
-rw--w---- 1 researcher2 research_team   46 Jul 26 08:49 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 26 08:49 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jul 26 08:49 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 26 08:49 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 26 08:49 project_r.txt
-rw-r--r-- 1 researcher2 research_team   46 Jul 26 08:49 project_t.txt
researcher2@794a61d0f2ab:~/projects$
```

Scenario:
The organization does not allow others to have write access to any files. Based on the current permissions shown in the previous step, we need to remove write access from others for any file that currently grants it.

From the output, we can see that the file project_t.txt has the following permission string:

-rw-rw-r--

This string indicates:

- **Owner**: Read + Write
- **Group**: Read + Write
- **Others**: Read only (write is already not allowed)

Although others do not have write access, the group does — which violates the policy if only the owner should have write permissions.

Command used:

```
chmod g-w,o-w project_t.txt
```

Updated permissions:

-rw-r--r-- 1 researcher2 research_team 46 Jul 26 08:49 project_t.txt

Explanation:

- The `chmod g-w,o-w` command removes **write access** from both the **group** and **others** for the file project_t.txt.
- This ensures **only the file owner** has permission to write or modify the file, while others can only read it.

Change file permissions on a hidden file

Scenario:
The research team has archived the file named .project_x.txt, which is why it's a hidden file (files beginning with a dot . are hidden in Linux). According to policy, no one should have **write permissions** on this file. However, both the **user** and **group** should be able to **read** it.

The original permission string was:

-rw--w----

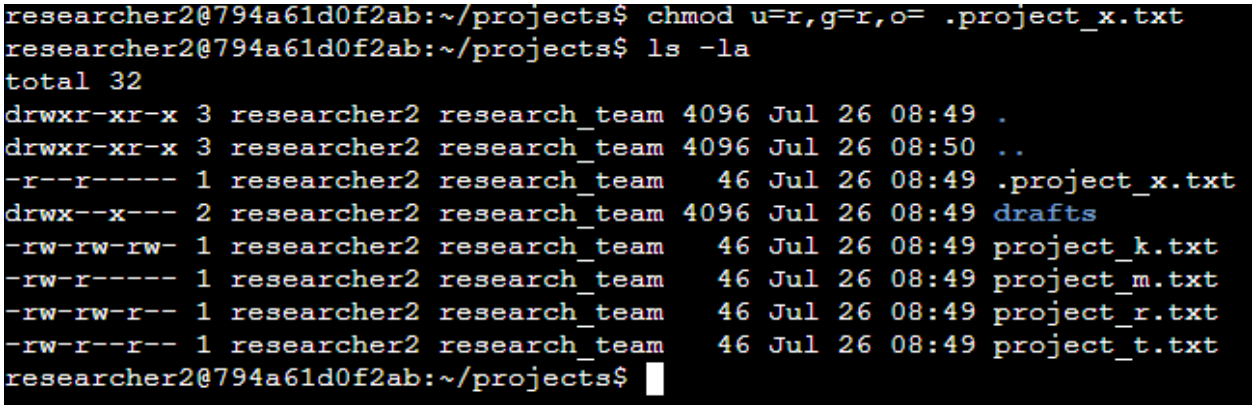
This indicates:

- **Owner**: Read and Write (Write should be removed)

- **Group**: Write only (Should be changed to Read only)
- **Others**: No access (Correct)

🔧 **Command used:**

```
chmod u=r,g=r,o= .project_x.txt
```



Updated permissions:

-r--r----- 1 researcher2 research_team 46 Jul 26 08:49 .project_x.txt

Explanation:
The command `chmod u=r,g=r,o=` performs the following actions:

- Grants **read-only access** to the **user** (u=r)
- Grants **read-only access** to the **group** (g=r)
- Removes **all permissions** for **others** (o=)
- Ensures that **no one** has write access, protecting the **integrity** of the archived hidden file

This aligns with organizational security policies for archived data.

Change directory permissions

Scenario:
Within the `projects` directory, there's a subdirectory named `drafts`. This folder contains confidential content and must only be accessed by the owner, `researcher2`. According to policy, no other team members or users should have permission to read, write, or enter this directory.

Currently, the permission string is:

`drwxr-xr-x`

This indicates:

- The user (owner) has permission to read, write, and enter the directory.
- The group can read and enter, but not write.
- Others can also read and enter, but not write.

This level of access violates the policy because it allows more users than intended to enter or view the contents of the directory.

Command used:

`chmod u=rwx,g=,o= drafts`

```
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$
```

Updated permissions:

```
drwx------ researcher2 research_team ... drafts
```

Explanation:

The `chmod u=rwx,g=,o=` command sets the following:

- **User (u)**: full permissions (read, write, execute)
- **Group (g)**: no permissions
- **Others (o)**: no permissions

This change **restricts all access** to the drafts directory **only to the owner**, effectively blocking others from viewing, modifying, or entering the directory. It fully complies with the **organization's confidentiality policy**.

Summary

In this project, I audited and modified file and directory permissions within a shared Linux environment to meet organizational security policies. I started by reviewing the current permission settings using `ls -la`, including for hidden files. Based on this audit, I applied permission changes using the `chmod` command to ensure that only authorized users and groups retained access, while unnecessary write permissions for group and others were removed.

I also secured sensitive and archived files such as `.project_x.txt` by granting read-only access to the user and group, while denying all access to others. Additionally, I enforced stricter privacy on the `drafts/` directory by allowing full access only to the owner (`researcher2`) and revoking access for all other users. These changes support the principle of least privilege and showcase effective permission management using Linux command-line tools.