File permissions in Linux

Project description

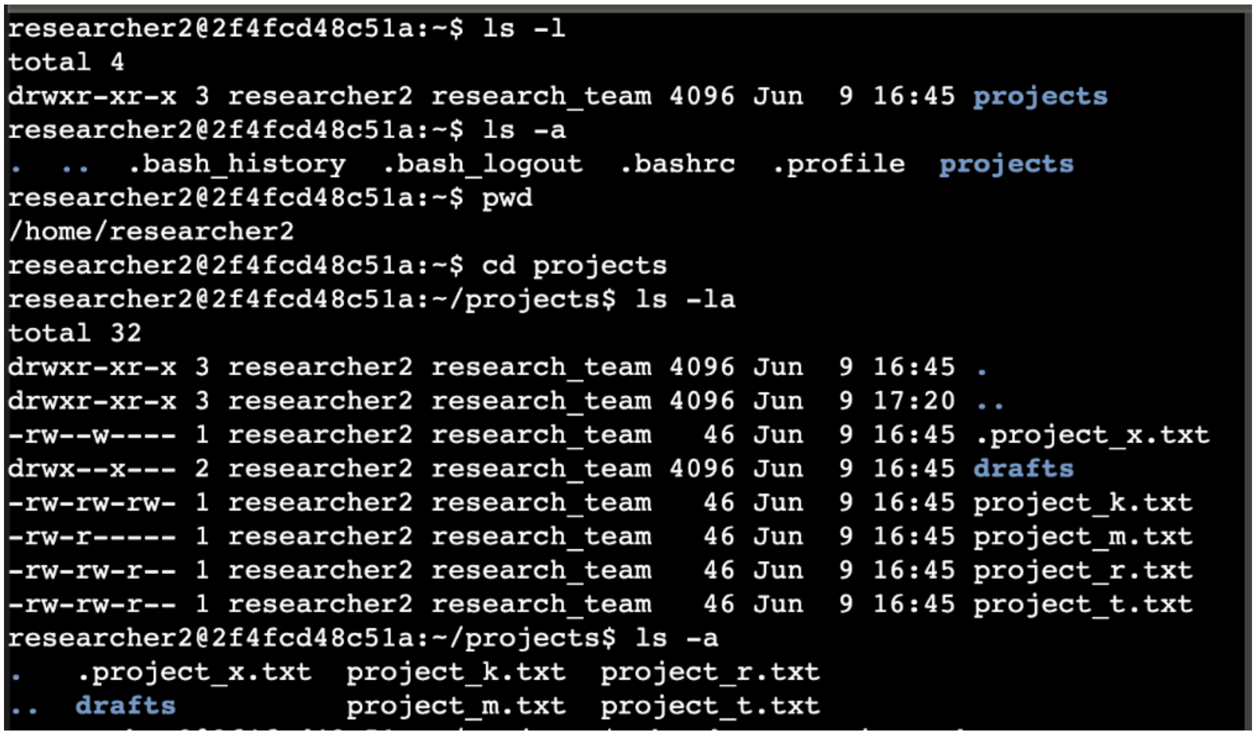
In this lab, I used Linux commands to examine and manage file and directory permissions within the `/home/researcher2/projects` directory. The goal was to ensure proper authorization by restricting access to files and directories according to the principle of least privilege. I identified files and directories with incorrect permissions, including hidden files, and modified them to remove unauthorized write and execute access for group and other users. This involved exploring file types, ownership, and permission strings, then applying `chmod` commands to tighten security on sensitive files and directories, such as `.project_x.txt` and the `drafts` directory. The exercise emphasized the importance of granular access control to prevent unauthorized access and protect sensitive information on a Linux system.

Check file and directory details

Objective:

Examine the files in the `/home/researcher2/projects` directory to:

- Check file and directory permissions
- Identify hidden files
- Understand ownership settings



```
researcher2@2f4fcd48c51a:~$ ls -l
total 4
drwxr-xr-x 3 researcher2 research_team 4096 Jun  9 16:45 projects
researcher2@2f4fcd48c51a:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .profile  projects
researcher2@2f4fcd48c51a:~$ pwd
/home/researcher2
researcher2@2f4fcd48c51a:~$ cd projects
researcher2@2f4fcd48c51a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jun  9 16:45 .
drwxr-xr-x 3 researcher2 research_team 4096 Jun  9 17:20 ..
-rw--w---- 1 researcher2 research_team   46 Jun  9 16:45 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jun  9 16:45 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jun  9 16:45 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jun  9 16:45 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun  9 16:45 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun  9 16:45 project_t.txt
researcher2@2f4fcd48c51a:~/projects$ ls -a
.   .project_x.txt  project_k.txt  project_r.txt
..  drafts          project_m.txt  project_t.txt
```

The **group that owns the files is:** `research_team`
When I ran `ls -la`, I checked for hidden files and found: `.project_x.txt`

Summary:

- I confirmed file ownership and permissions.
- I identified that `project_k.txt` had world-writable permissions, which is a security risk. I fixed this in the next task.

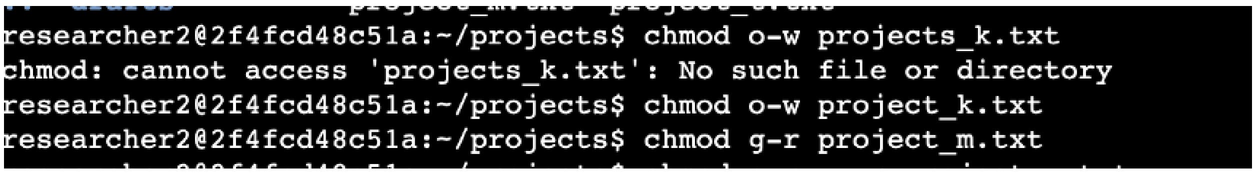
Explanation :

- `cd` changes the working directory so I can work inside `projects`.
- `ls -l` shows detailed file information like permissions, ownership, and timestamps.
- `ls -la` lists **all files**, including hidden ones (those that start with a dot `.`).

Change file permissions

Objective:

Identify files with incorrect permissions and fix them to remove unauthorized access. Specifically, ensure **no files are writable by "other" users** and that sensitive files have **tighter access control.**



```
researcher2@2f4fcd48c51a:~/projects$ chmod o-w projects_k.txt
chmod: cannot access 'projects_k.txt': No such file or directory
researcher2@2f4fcd48c51a:~/projects$ chmod o-w project_k.txt
researcher2@2f4fcd48c51a:~/projects$ chmod g-r project_m.txt
```

Findings:

File with incorrect permissions:

- `project_k.txt` was world-writable (`rw-rw-rw-`), which is a security risk because **any user could modify it.**
- Fixed it using `chmod o-w project_k.txt`.

Sensitive file (project_m.txt) issue:

- Group had read access (`rw-r-----`), but **this file should only be accessible by the file owner.**
- Removed group permissions using `chmod g-rw project_m.txt`.

New Permissions:

- `project_k.txt` → `rw-r--r--` ✔️ (others can no longer write)
- `project_m.txt` → `rw-------` ✔️ (only the user has access)

Explanation:

- `chmod o-w project_k.txt` → Takes **write permission away from "others"** (anyone not the file owner or group).
- `chmod g-rw project_m.txt` → Takes **read and write permissions away from the group** (only the owner can access now).
- Running `ls -l` again confirms the permissions were changed correctly.

Change file permissions on a hidden file

Objective:

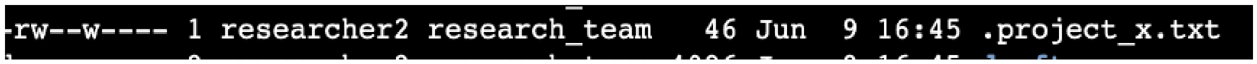
The goal of this task is to secure the hidden file `.project_x.txt` by ensuring that **no one can write to it,** but the **user and group should still be able to read it.** This helps protect archived files from accidental changes or unauthorized modifications.

Findings:

When I checked the permissions of the hidden file using:

```
ls -la
```

I found:



```
-rw--w---- 1 researcher2 research_team   46 Jun  9 16:45 .project_x.txt
```

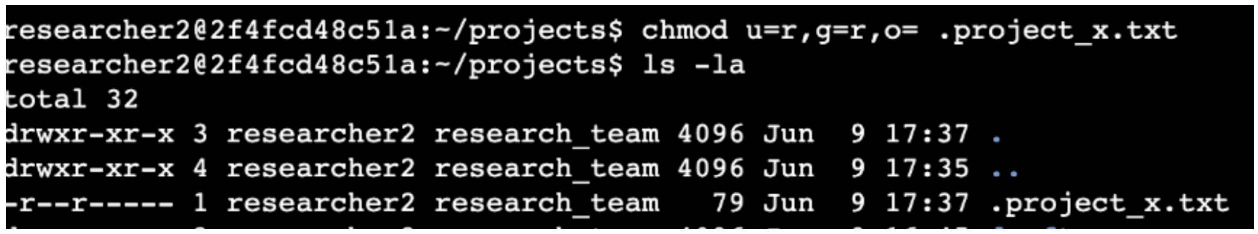
✔️ **Problem:**

- **User and Group have WRITE access, which is NOT okay.**
- This file is archived, so **nobody should be able to write to it.**

✔️ **Which owner type had incorrect permissions?**
👉 **Both the user and the group had incorrect write permissions.**

What I Changed:

- **Before:** `-rw--w----`
- **After:** `-r--r-----`



```
researcher2@2f4fcd48c51a:~/projects$ chmod u=r,g=r,o= .project_x.txt
researcher2@2f4fcd48c51a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jun  9 17:37 .
drwxr-xr-x 4 researcher2 research_team 4096 Jun  9 17:35 ..
-r--r----- 1 researcher2 research_team   79 Jun  9 17:37 .project_x.txt
```

Command I Used:

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

Explanation:

- chmod → Command to change file permissions.
- u=r → User gets only read permission.
- g=r → Group gets only read permission.
- o= → Others get no permissions at all.
- .project_x.txt → The hidden file I'm securing.

Change directory permissions

Objective:

The goal of this task was to **lock down the `drafts` directory** so that **only the `researcher2` user can access it.** The group and others should have **no permissions at all.**

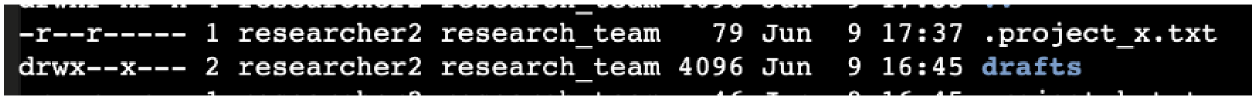
Problem:

- The **group still had execute (x) permissions.**
- Execute permission on a directory = **the ability to access and list files inside it.**
- Only `researcher2` should be able to do that.

✔️ **Does the group have permissions to access the drafts directory?**
👉 **Yes.**

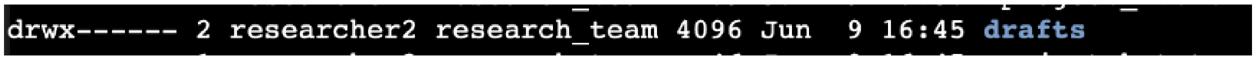
What I Changed:

- **Before:** `drwx--x---`



```
-r--r----- 1 researcher2 research_team   79 Jun  9 17:37 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jun  9 16:45 drafts
```

- **After:** `drwx------`



```
drwx------ 2 researcher2 research_team 4096 Jun  9 16:45 drafts
```

Command I Used:

```
chmod g-x drafts
```

Explanation:

- `chmod` ⟶ Used to change file or directory permissions.
- `g-x` ⟶ Remove execute permission from the group.
- `drafts` ⟶ This is the directory I was securing.

✔️ Why this matters:

- Execute permission on a directory = ability to “enter” or list its files.
- By removing this, group members can no longer access or browse the drafts directory.
- Now, only the user (researcher2) can enter and access it.

Verification:
After running the command:

```
ls -ld drafts
```

I confirmed the new permissions

Why This Is Important:

- Locking down sensitive directories is crucial for preventing unauthorized access.
- Even if a group member is part of the research_team, they shouldn’t access drafts without explicit permission.
- This command helps enforce principle of least privilege — give people only the access they truly need.

Summary

- Verified file and directory permissions using `ls -la` and `ls -ld` to check both visible and hidden files.
- Identified `project_k.txt` as world-writable (`rw-rw-rw-`), posing a security risk; removed write permission from others using `chmod o-w`.
- Found that `project_m.txt` was accessible by the group but needed to be restricted to the user only; removed group read and write permissions using `chmod g-rw`.
- Secured the hidden archived file `.project_x.txt` by removing write permissions from both user and group, leaving only read access, via `chmod u=r,g=r,o=`.
- Locked down the `drafts` directory to restrict access solely to the owner (researcher2) by removing group execute permission using `chmod g-x`.

- Confirmed all permission changes by re-listing directory contents and permissions.
- This process enforced strict authorization, ensuring only authorized users can access or modify sensitive files and directories, aligning with security best practices.