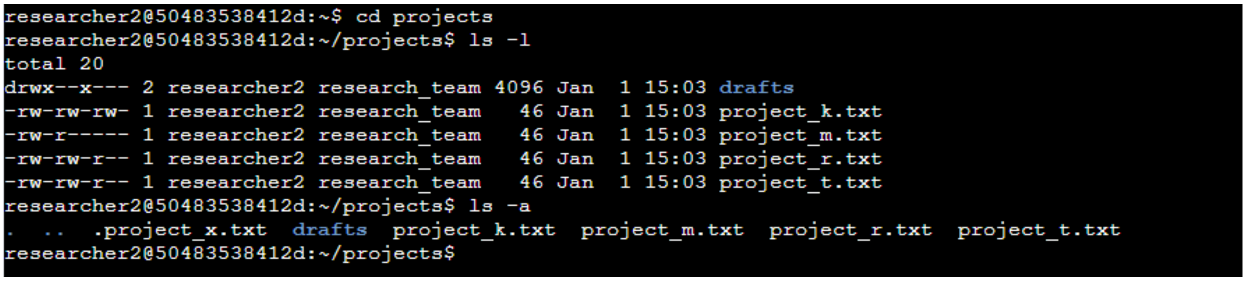
File permissions in Linux

Project description

In this project, I worked in a Linux environment to secure files and folders by fixing incorrect permissions. The goal was to make sure that only the right users could access sensitive data. This helps prevent unauthorized access, data leaks, and accidental file changes.

I used common Linux commands like `ls`, `chmod`, and `cd` to review and fix file and directory permissions.

Check file and directory details



```
researcher2@50483538412d:~$ cd projects
researcher2@50483538412d:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan  1 15:03 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan  1 15:03 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan  1 15:03 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_t.txt
researcher2@50483538412d:~/projects$ ls -a
.  ..  .project_x.txt  drafts  project_k.txt  project_m.txt  project_r.txt  project_t.txt
researcher2@50483538412d:~/projects$
```

First, I moved into the `projects` folder and listed all files using ls -l. This command shows who owns each file and what permissions they have. I confirmed that the files belonged to the **research_team** group and checked for hidden files using: ls -a This revealed a hidden file called `.project_x.txt`, which also needed to be secured.

Describe the permissions string

In Linux, a permissions string is a sequence of 10 characters that defines who can access a file or directory and what they are allowed to do with it. You can see these strings by running the ls -l command in your terminal.

How to read the String

The string is broken down into four distinct parts:

Position	Part	Description

1st Char	**Type**	- for a file, d for a directory.
Chars 2-4	**Owner**	What the creator/owner can do.
Chars 5-7	**Group**	What members of the file group can do.
Chars 8-10	**Others**	What everyone else on the system can do.

The Three Symbols

Inside those sets of three, you will see these letters:

r **(Read):** Permission to view the file contents or list a directory.

w **(Write):** Permission to modify the file or add/remove files in the directory.

x **(Execute):** Permission to run a file as a program or “enter” a directory.

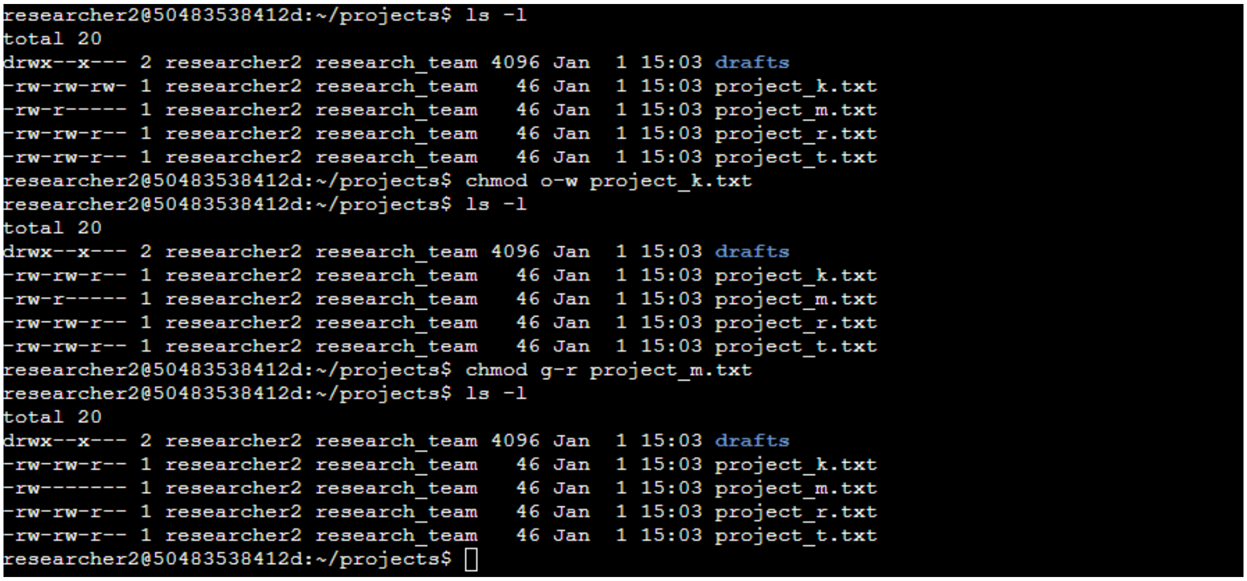
- **(None):** Indicates that specific permission is denied.

Examples:

If you see -rwxr-xr-- :

1. It’s a regular file
2. The owner can read, write and execute the file.
3. The group can read and execute but not write.
4. Others can only read the file.

Change file permissions

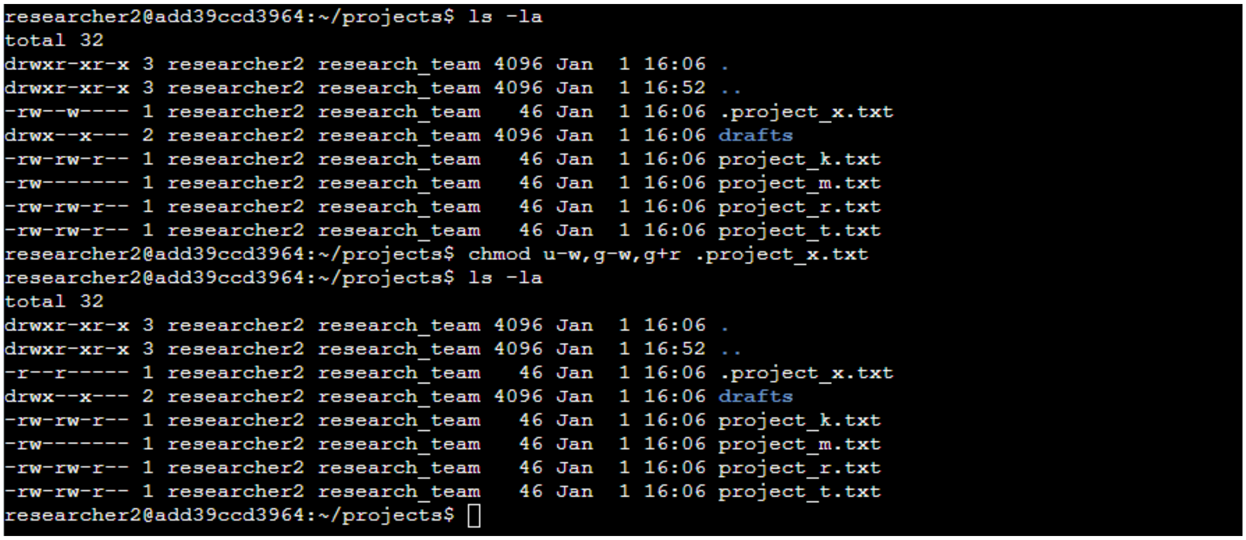


```
researcher2@50483538412d:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan  1 15:03 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan  1 15:03 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan  1 15:03 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_t.txt
researcher2@50483538412d:~/projects$ chmod o-w project_k.txt
researcher2@50483538412d:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan  1 15:03 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan  1 15:03 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_t.txt
researcher2@50483538412d:~/projects$ chmod g-r project_m.txt
researcher2@50483538412d:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan  1 15:03 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_k.txt
-rw------- 1 researcher2 research_team   46 Jan  1 15:03 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 15:03 project_t.txt
researcher2@50483538412d:~/projects$ 
```

Some files had unsafe permissions that allowed **others** to write to them. This is dangerous because anyone on the system could change or delete those files.

For example, the file `project_k.txt` had write access for **others**, so I removed it using: chmod o-w project_k.txt This made sure only authorized users could modify the file. The file `project_m.txt` contained sensitive data. It was supposed to be accessible only by the owner, but the group had read access. I verified this with: ls -l then I removed the group's read permission: chmod g-r project_m.txt Now, only the owner can read or modify this file, which protects it from unauthorized access.

Change file permissions on a hidden file

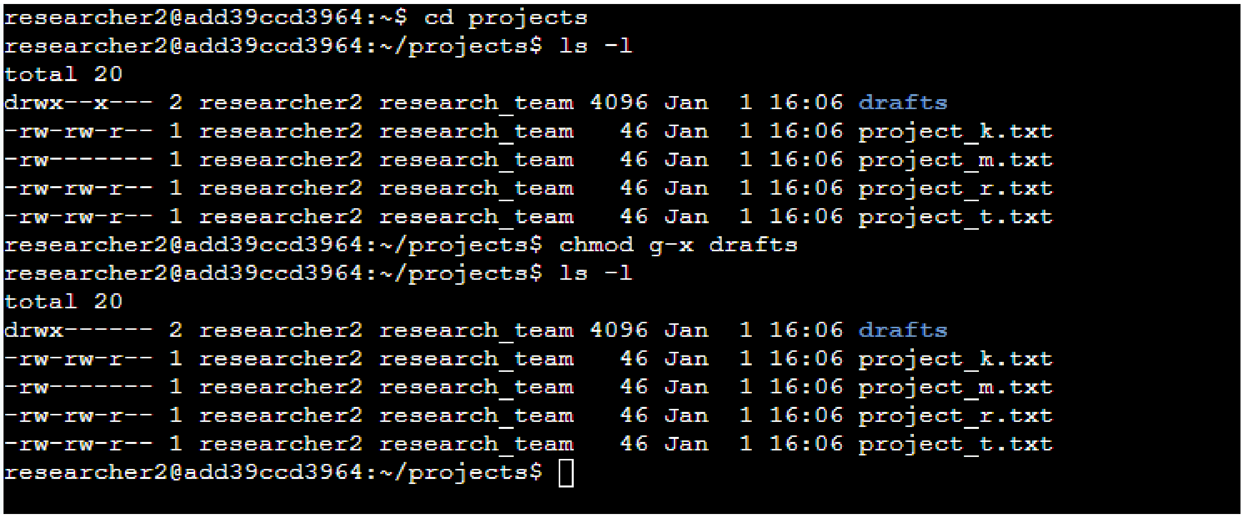


```
researcher2@add39ccd3964:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan  1 16:06 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan  1 16:52 ..
-rw--w---- 1 researcher2 research_team   46 Jan  1 16:06 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan  1 16:06 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_k.txt
-rw------- 1 researcher2 research_team   46 Jan  1 16:06 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_t.txt
researcher2@add39ccd3964:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@add39ccd3964:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan  1 16:06 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan  1 16:52 ..
-r--r----- 1 researcher2 research_team   46 Jan  1 16:06 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan  1 16:06 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_k.txt
-rw------- 1 researcher2 research_team   46 Jan  1 16:06 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_t.txt
researcher2@add39ccd3964:~/projects$ 
```

Hidden files can be risky because they are easy to overlook. I checked the permissions of `.project_x.txt` and found it was too open. I fixed this using: chmod u-w,g-w,g+r .project_x.txt

This ensured that the user and the group can only read the hidden file and no one else.

Change directory permissions



```
researcher2@add39ccd3964:~$ cd projects
researcher2@add39ccd3964:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan  1 16:06 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_k.txt
-rw------- 1 researcher2 research_team   46 Jan  1 16:06 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_t.txt
researcher2@add39ccd3964:~/projects$ chmod g-x drafts
researcher2@add39ccd3964:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Jan  1 16:06 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_k.txt
-rw------- 1 researcher2 research_team   46 Jan  1 16:06 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan  1 16:06 project_t.txt
researcher2@add39ccd3964:~/projects$ 
```

Directories also need correct permissions. I checked the `drafts` directory inside the `projects` folder and found that the group had execute permission, which allows them to enter the directory. I removed it using: chmod g-x drafts This prevented unauthorized users from browsing or accessing the contents of the folder.

Summary

By fixing these permissions, I secured:

- Normal files
- Hidden files
- Sensitive data
- Project directories

I applied the **principle of least privilege**, meaning each user only has the access they actually need. This reduces the risk of data leaks, accidental changes, and insider threats.

What This Shows

This project demonstrates that I can:

- Read and understand Linux permission strings
- Find insecure file and folder permissions
- Use Linux commands to fix security issues
- Protect data in a multi-user system