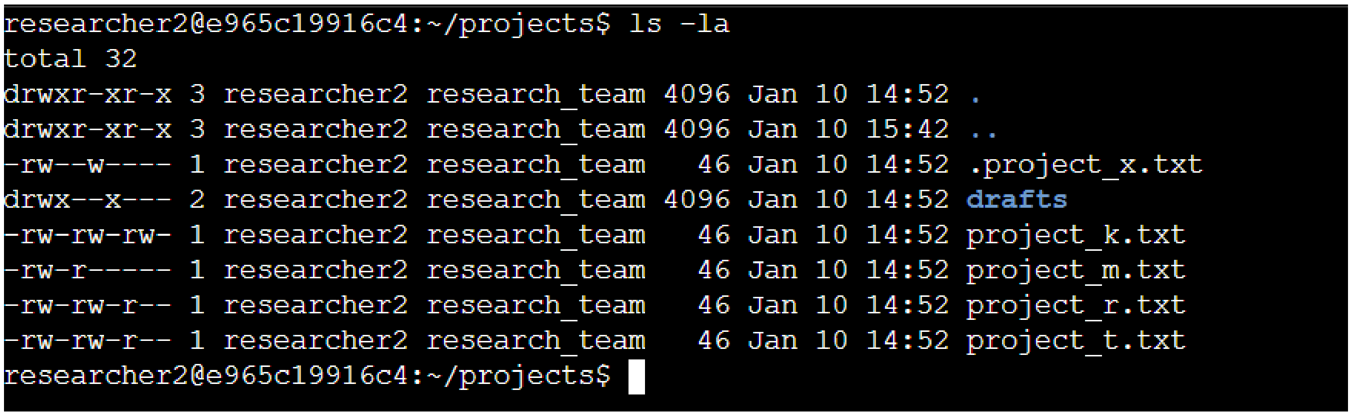
File permissions in Linux

Project description

As a security professional working with a research team, I needed to update the file permissions for certain files and directories within the `projects` directory. The existing permissions did not reflect the appropriate level of authorization. Checking and updating these permissions was essential to maintain the security of the system.

Check file and directory details

To determine the existing permissions set for the directory, I used the following Linux command:



```
researcher2@e965c19916c4:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 14:52 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 15:42 ..
-rw--w---- 1 researcher2 research_team   46 Jan 10 14:52 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan 10 14:52 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan 10 14:52 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 10 14:52 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_t.txt
researcher2@e965c19916c4:~/projects$
```

I used the `ls` command with the `-la` option to display a detailed listing of all files, including hidden files. The output indicates there is one directory named `drafts`, one hidden file named `.project_x.txt`, and five other project files.

Describe the permissions string

The 10-character string in the first column represents the permissions set on each file or directory. This string can be deconstructed to determine authorization:

- **1st character**: Indicates the file type; a `d` represents a directory, and a hyphen (`-`) represents a regular file.
- **2nd-4th characters**: Represent read (`r`), write (`w`), and execute (`x`) permissions for the **user**.
- **5th-7th characters**: Represent read (`r`), write (`w`), and execute (`x`) permissions for the **group**.

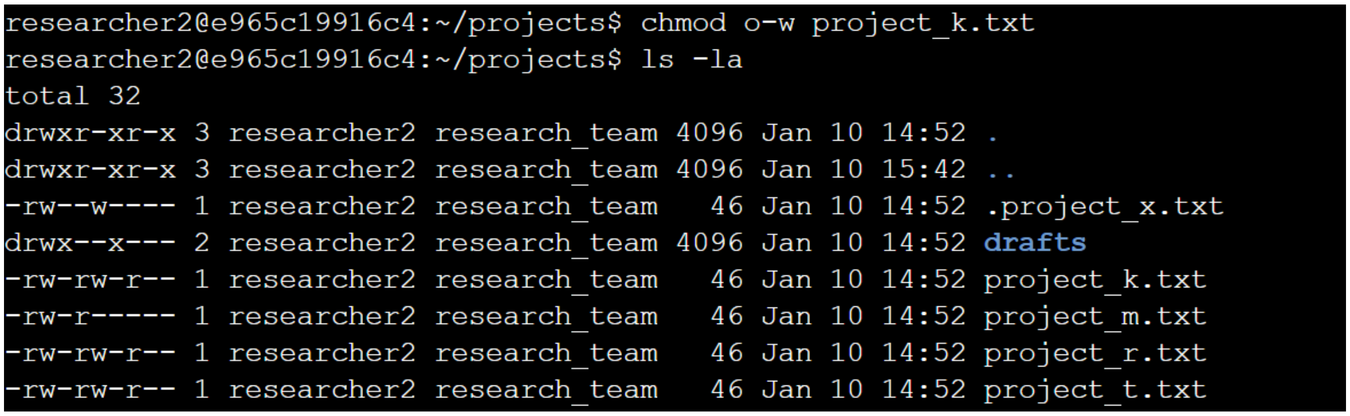
- **8th-10th characters**: Represent read (`r`), write (`w`), and execute (`x`) permissions for **others**.

For example, the file `project_t.txt` has the permissions `-rw-rw-r--`. This indicates it is a regular file where the user and group have read and write permissions, but others only have read permissions.

Change file permissions

The organization determined that "others" should not have write access to any files. Based on my previous check, I determined that `project_k.txt` needed to have write access removed for others.

I used the following command to modify the permissions:



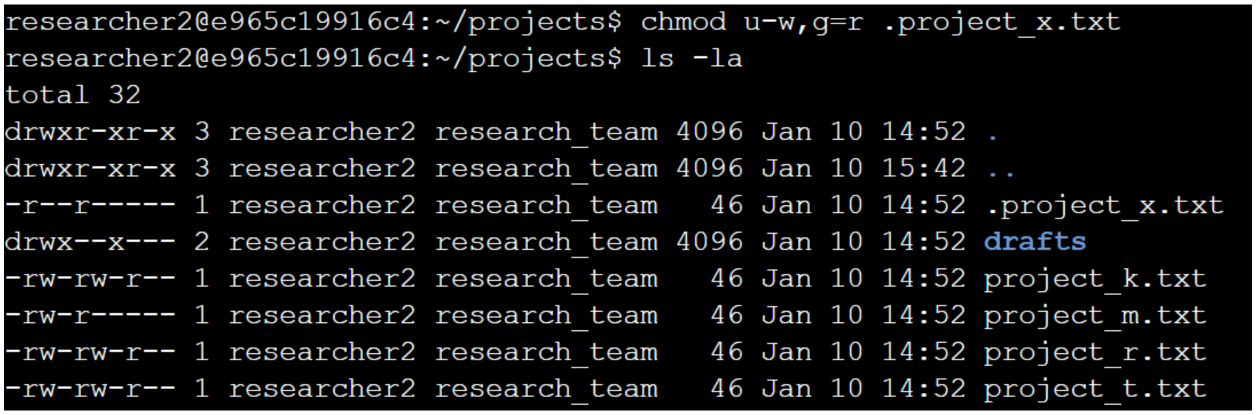
```
researcher2@e965c19916c4:~/projects$ chmod o-w project_k.txt
researcher2@e965c19916c4:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 14:52 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 15:42 ..
-rw--w---- 1 researcher2 research_team   46 Jan 10 14:52 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan 10 14:52 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 10 14:52 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_t.txt
```

The `chmod` command was used with `o-w` to remove write permissions from the "others" category for that specific file.

Change file permissions on a hidden file

The research team archived `.project_x.txt` and requested that no one have write access, but the user and group should retain read access. I identified it as a hidden file because its name starts with a period.

I used the following command to change the permissions:



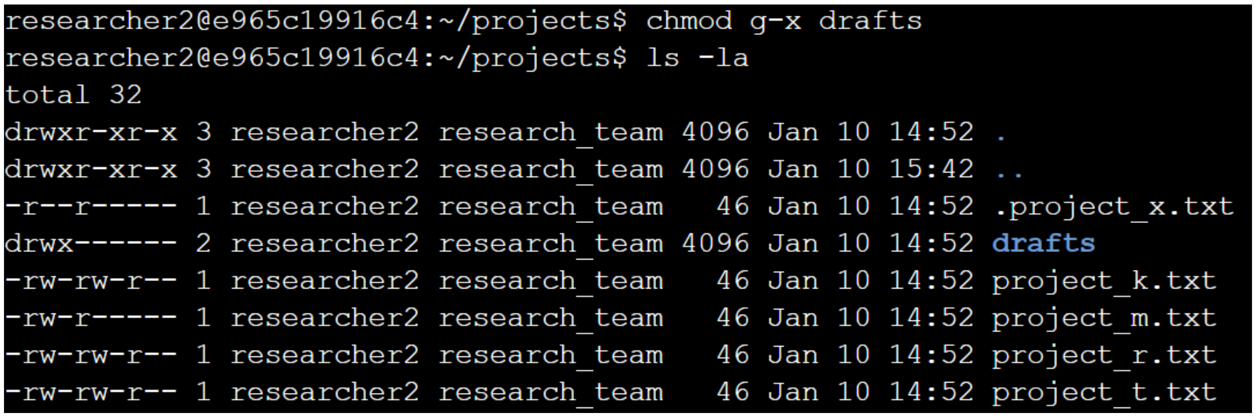
```
researcher2@e965c19916c4:~/projects$ chmod u-w,g=r .project_x.txt
researcher2@e965c19916c4:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 14:52 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 15:42 ..
-r--r----- 1 researcher2 research_team   46 Jan 10 14:52 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan 10 14:52 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 10 14:52 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_t.txt
```

In this step, I removed write permissions from the user (`u-w`), and ensured the group had only read permissions (`g=r`).

Change directory permissions

The organization requires that only the `researcher2` user has access to the `drafts` directory. This means no one else, including the group, should have execute permissions.

I used the following command to restrict access:



```
researcher2@e965c19916c4:~/projects$ chmod g-x drafts
researcher2@e965c19916c4:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 14:52 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 10 15:42 ..
-r--r----- 1 researcher2 research_team   46 Jan 10 14:52 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Jan 10 14:52 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 10 14:52 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 10 14:52 project_t.txt
```

By using `chmod g-x`, I removed the execute permissions that the group previously held, leaving only the user with execute access to the directory.

Summary

I changed multiple permissions to match the level of authorization required by the organization for files and directories in the `projects` directory. I began by using `ls -la` to audit the current permissions, which informed my subsequent decisions. I then used the `chmod` command multiple times to correct the permissions for files, hidden files, and directories to ensure the system remained secure.