File permissions in Linux (My Responses)

Project description

The research team at my organization needs to update file permissions for certain files and directories within the *projects* directory to ensure proper authorization and maintain system security. In this activity, I will use Linux commands to configure these file and directory permissions. Authorization is crucial because, without proper access control, users could modify or view files they shouldn't, which poses significant security risks. In Linux, file and directory permissions define who can read, write, or execute specific files, and it's essential to manage these permissions carefully to protect sensitive data. As a security analyst, it's my responsibility to ensure that the file permissions are correctly configured to safeguard the system. In this task, I'll be examining and managing the permissions for files within the /home/researcher2/projects directory, which belong to the researcher2 user and the research_team group. The goal is to check the permissions for all files, including hidden files, to ensure they align with the necessary access levels. If any permissions are incorrect, I will adjust them accordingly. I will also ensure that the /home/researcher2/projects/drafts directory is properly secured by modifying its permissions to remove unauthorized access. By completing this task, I will gain hands-on experience in managing file permissions in Linux, ensuring that only authorized users can access sensitive information and keeping the system secure from potential threats.

Check file and directory details

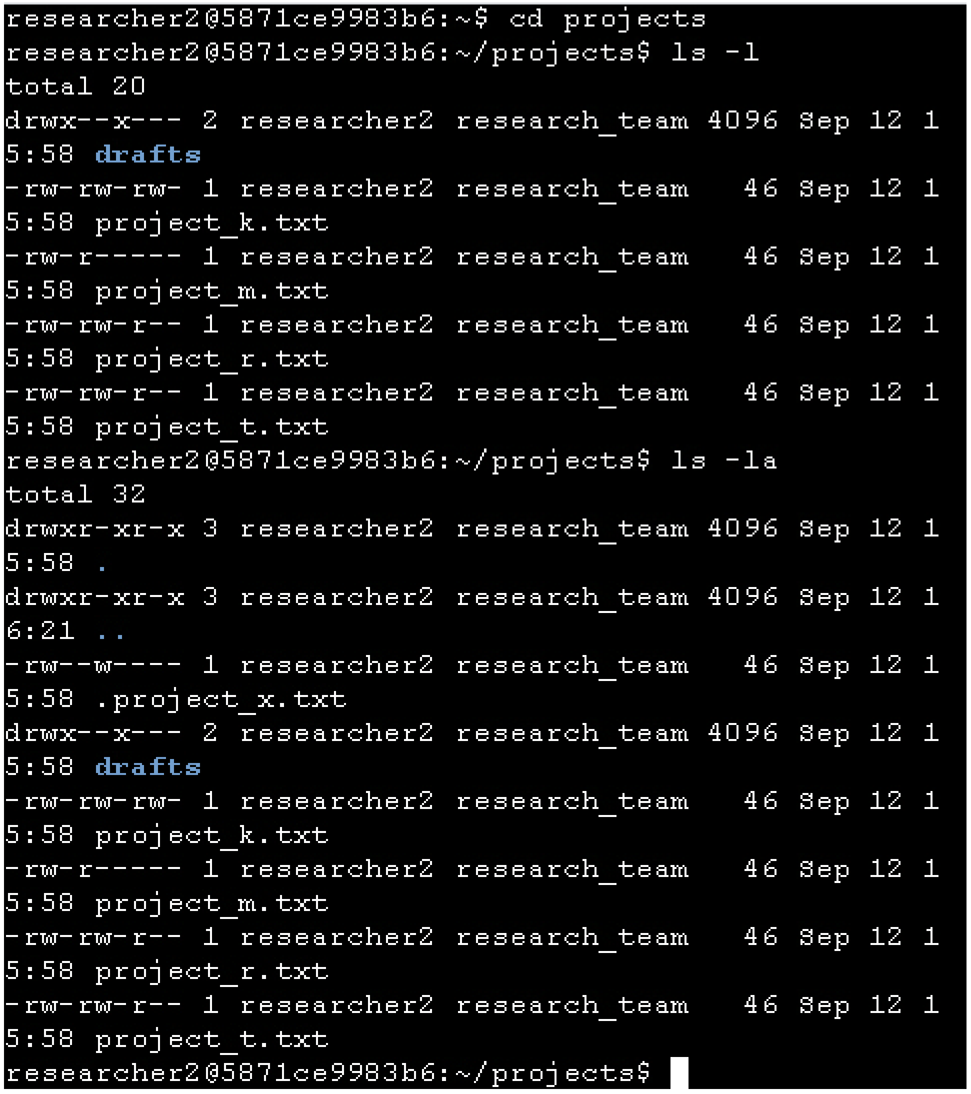
The goal of this task was to examine the permissions of the files and directories in the *projects* directory, which belongs to the researcher2 user. By using the cd command to navigate to the directory and the ls -l command to list the contents along with their permissions, I was able to check if any files had the wrong access settings. The permissions were displayed in a 10-character string, indicating whether a file or directory allowed read, write, or execute actions for the user, group, or others. I learned that the group owning the files in the *projects* directory was research_team. Additionally, I checked for hidden files using the ls -la command and found that .project_x.txt was hidden. This gave me an understanding of how to work with both visible and hidden files.

Solution: The ls -l and ls -la commands helped identify permissions and hidden files.

The following code demonstrates how I used Linux commands to determine the existing permissions set for a specific directory in the file system.

```
cd projects
ls -l
```
OR
```
cd projects
ls -la
```

Q & A: What is the name of the group that owns the files in the projects directory? The research_team owns the files in the projects directory. Which of these files is hidden in the projects directory? The .project_x.txt file is hidden.



```
researcher2@5871ce9983b6:~$ cd projects
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Sep 12 1
5:58 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 12 1
5:58 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 12 1
5:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 1
5:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 1
5:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 1
5:58 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 1
6:21 ..
-rw--w---- 1 researcher2 research_team   46 Sep 12 1
5:58 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 12 1
5:58 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 12 1
5:58 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 12 1
5:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 1
5:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 1
5:58 project_t.txt
researcher2@5871ce9983b6:~/projects$
```

The first line of the screenshot displays the command I entered, and the other lines display the output. The code lists all contents of the `projects` directory. I used the `ls` command with the `-la` option to display a detailed listing of the file contents that also returned hidden files. The output of my command indicates that there is one directory named `drafts`, one hidden file named `.project_x.txt`, and five other project files. The 10-character string in the first column represents the permissions set on each file or directory.

Describe the permissions string

The 10-character string can be deconstructed to determine who is authorized to access the file and their specific permissions. The characters and what they represent are as follows:

- **1st character**: This character is either a `d` or hyphen (`-`) and indicates the file type. If it's a `d`, it's a directory. If it's a hyphen (`-`), it's a regular file.
- **2nd-4th characters**: These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the user. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted to the user.
- **5th-7th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the group. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted for the group.
- **8th-10th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for other. This owner type consists of all other users on the system apart from the user and the group. When one of these characters is a hyphen (`-`) instead, that indicates that this permission is not granted for other.

For example, the file permissions for `project_t.txt` are `-rw-rw-r--`. Since the first character is a hyphen (`-`), this indicates that `project_t.txt` is a file, not a directory. The second, fifth, and eighth characters are all r, which indicates that user, group, and other all have read permissions. The third and sixth characters are `w`, which indicates that only the user and group have write permissions. No one has execute permissions for `project_t.txt`.

Change file permissions

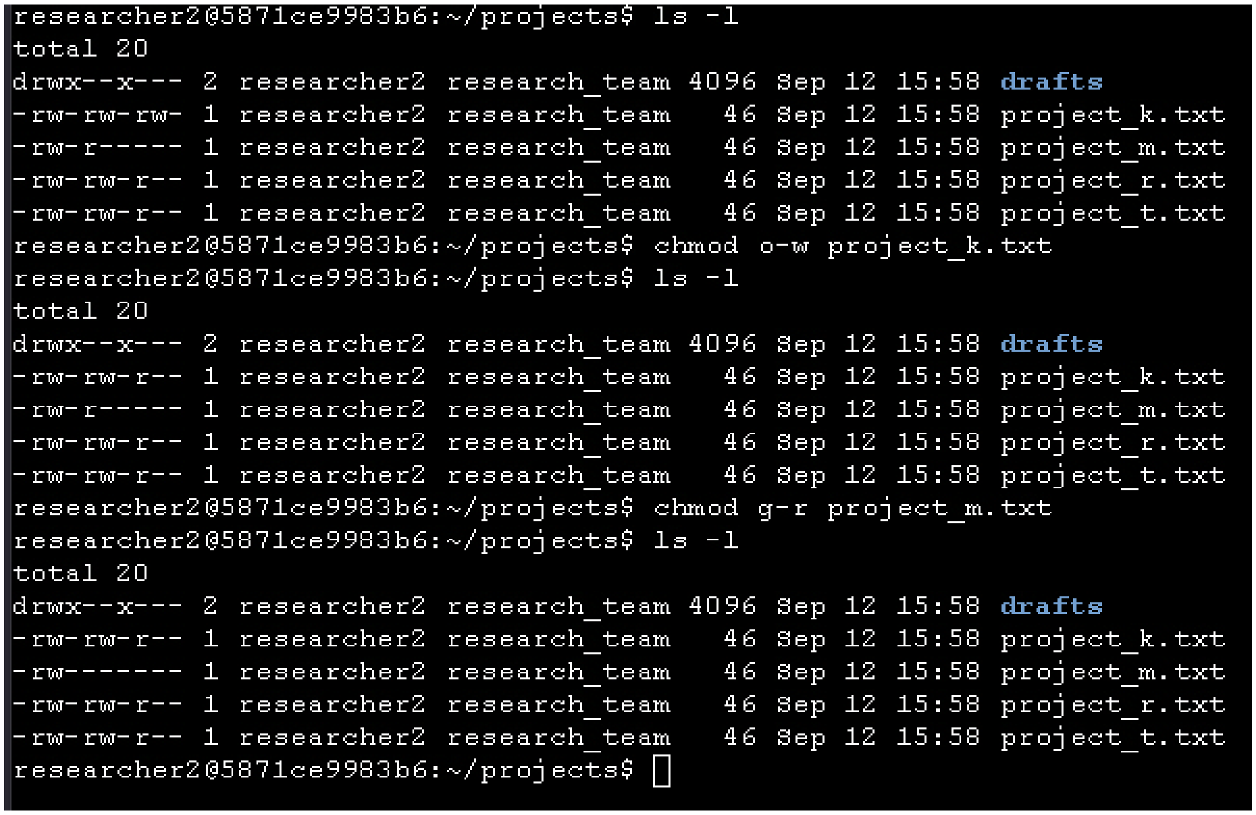
The goal of this task was to ensure that files in the *projects* directory had correct permissions and no unauthorized users could access or modify files. I started by identifying that the file project_k.txt had write permissions for others, which posed a security risk. To resolve this, I used the chmod o-w project_k.txt command to remove the write permission for "others." Next, I focused on the file project_m.txt, which should only be accessible by the user, not the group or others. After checking its permissions using ls -l, I used the chmod g-r project_m.txt command to remove the group's read and write permissions.

Solution: The chmod o-w and chmod g-r commands were used to adjust the permissions appropriately, removing unauthorized access.

The following code demonstrates how I used Linux commands to do this:

```
cd projects
ls -l
chmod o-w project_k.txt
ls -l
chmod g-r project_m.txt
ls -l
```

Q & A: Which file grants other users write permissions? The project_k.txt file has write permissions for other users. What are the group permissions on the project_m.txt file? The group permissions of the project_m.txt file is read only.



```
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ chmod o-w project_k.txt
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ chmod g-r project_m.txt
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ 
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. The `chmod` command changes the permissions on files and directories. The first argument indicates what permissions should be changed, and the second argument specifies the file or directory. In this example, I removed write permissions from other for the `project_k.txt` file. After this, I used `ls -la` to review the updates I made.

Change file permissions on a hidden file

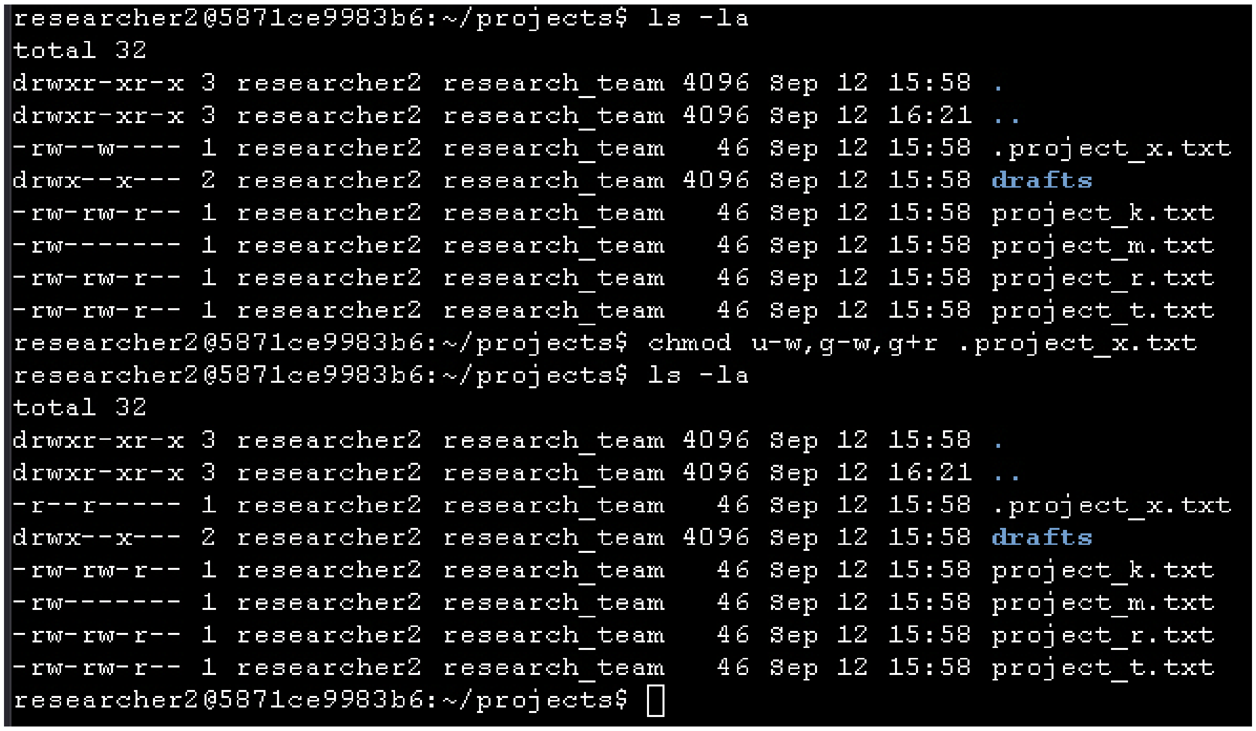
This task involved checking and modifying the permissions of a hidden file .project_x.txt. The file was archived and should not be written to by anyone, though it should still be readable by the user and group. Using the ls -la command, I found that both the user and group had write permissions, which was incorrect. To fix this, I used the chmod u-w,g-w,g+r .project_x.txt command to ensure that both the user and group could only read the file and not write to it.

Solution: The chmod u-w,g-w,g+r command helped adjust the permissions to ensure appropriate access levels for the user and group.

The following code demonstrates how I used Linux commands to change the permissions:

```
cd project
ls -la
chmod u-w,g-w,g+r .project_x.txt
ls -la
```

Q & A: Which owner type has the incorrect write permissions? The user and group owner types have incorrect write permissions.



```
researcher2@5871ce9983b6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 15:58 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 16:21 ..
-rw--w---- 1 researcher2 research_team   46 Sep 12 15:58 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@5871ce9983b6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 15:58 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 12 16:21 ..
-r--r----- 1 researcher2 research_team   46 Sep 12 15:58 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ 
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I know `.project_x.txt` is a hidden file because it starts with a period (`.`). In this example, I removed write permissions from the user and group, and added read permissions to the group. I removed write permissions from the user with `u-w`. Then, I removed write permissions from the group with `g-w`, and added read permissions to the group with `g+r`.

Change directory permissions

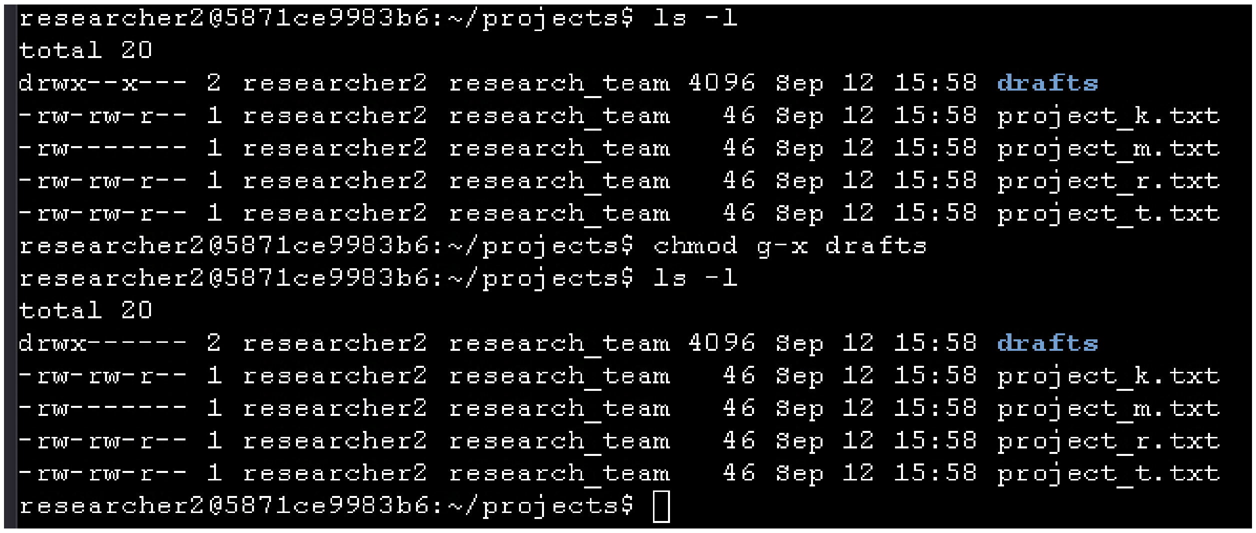
In this task, I focused on the *drafts* directory within the *projects* directory. The goal was to ensure that only the researcher2 user had access to this directory, preventing unauthorized access by others. Using the ls -l command, I discovered that the group had execute permissions, allowing them to access the directory. To secure the directory, I used the chmod g-x drafts command to remove the group's execute permissions, ensuring that only the researcher2 user could access it.

Solution: The chmod g-x command removed the execute permissions for the group on the drafts directory.

The following code demonstrates how I used Linux commands to change the permissions:

```
cd project
ls -l
chmod g-x drafts
```

Q & A: Does the group have permissions set to access the drafts directory and its contents? Yes, the group has execute permissions and therefore has access to the drafts directory.



```
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$ chmod g-x drafts
researcher2@5871ce9983b6:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Sep 12 15:58 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 12 15:58 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 12 15:58 project_t.txt
researcher2@5871ce9983b6:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I previously determined that the group had execute permissions, so I used the `chmod` command to remove them. The `researcher2` user already had execute permissions, so they did not need to be added.

Summary

This activity provided hands-on experience with managing file and directory permissions in Linux. By learning to change the Linux code commands like ls -l, ls -la, and chmod, I was able to check, modify, and secure file and directory permissions. These skills are essential for a security analyst, as all of these demo commands ensure that only authorized users have the appropriate level of access to sensitive files and directories. Overall, I enjoyed this task as it taught me how to enforce strict authorization controls to maintain the integrity and security of the system.