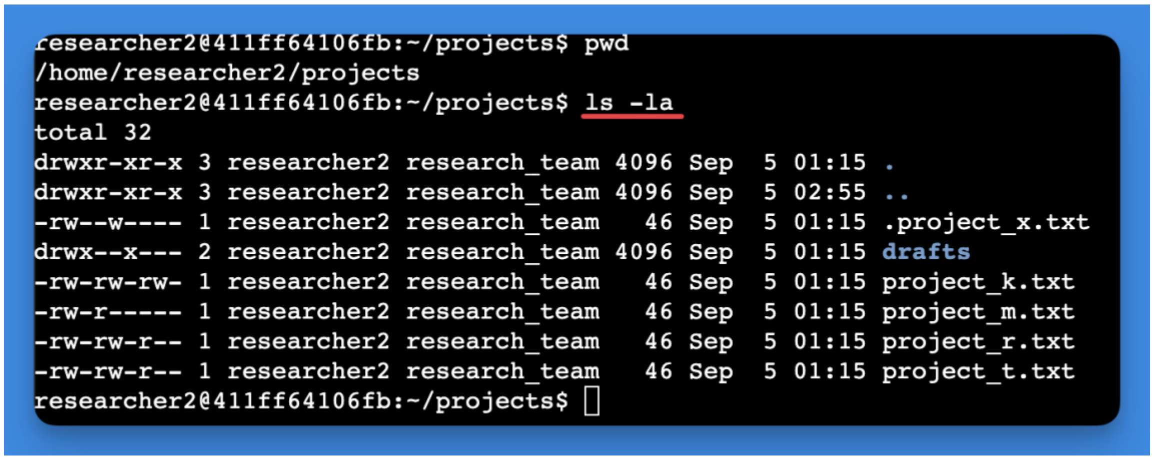
File permissions in Linux

Project description

In today's complex cybersecurity landscape, ensuring that file system permissions align with authorized user roles is critical for maintaining the integrity and security of an organization's data. As a security professional serving on an organization, the project focuses on auditing existing file system permissions and rectifying any discrepancies. The main objectives are to identify unauthorized access points and to align permissions with the appropriate user roles within the team.

Check file and directory details

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



```
researcher2@411ff64106fb:~/projects$ pwd
/home/researcher2/projects
researcher2@411ff64106fb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 01:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 02:55 ..
-rw--w---- 1 researcher2 research_team   46 Sep  5 01:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep  5 01:15 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep  5 01:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep  5 01:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_t.txt
researcher2@411ff64106fb:~/projects$ 
```

The screenshot begins with the command I executed, followed by its output. I utilized the `ls` command with the `-la` flag to generate a comprehensive list of all items in the `'projects`' directory, including hidden files. The output reveals the presence of one folder called `'drafts`,' a hidden file named `'.project_x.txt`,' and five additional project files. The string of 10 characters in the initial column signifies the permission set for each listed file or directory.

Describe the permissions string

A notable aspect of system file security is the 10-character string representing file and directory permissions in Linux. These characters can be dissected to comprehend the authorization level and specific permissions assigned to different classes of users.

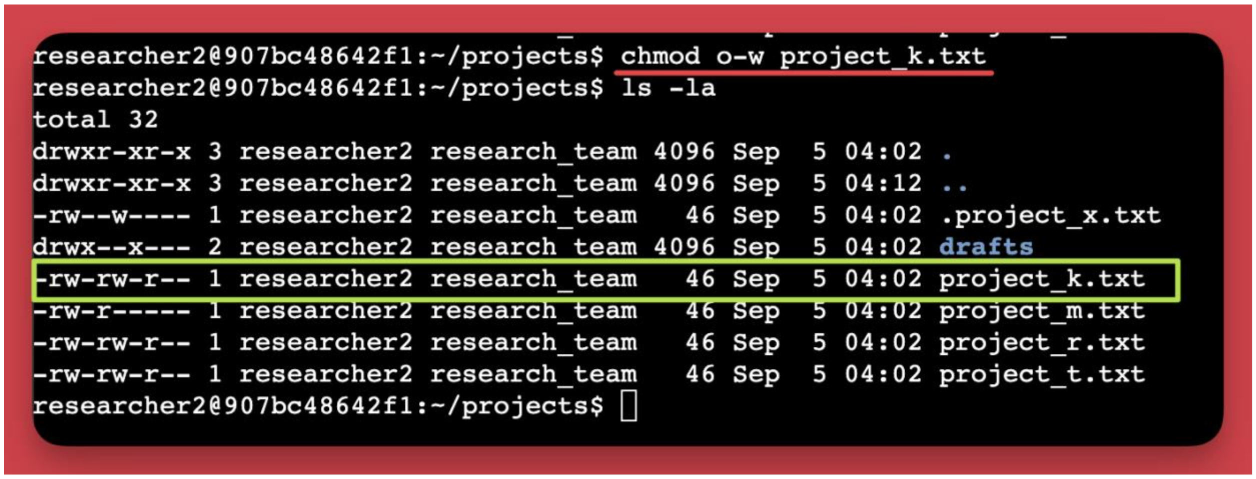
- **1st Character**: Represents the type of the item. A `'d'` signifies it's a directory, while a hyphen `'-'` indicates it's a regular file.
- **Characters 2-4:** Demonstrate the `'read (r)'`,`'write (w)'` , and `'execute (x)'` permissions for the file's owner. A hyphen `'-'` in any of these positions means that the specific permission isn't assigned to the owner .
- **Characters 5-7:** Represent the same set of permissions but for the group to which the file belongs. A hyphen here implies the corresponding permission is not enabled for the group.
- **Characters 8-10:** Outline the permissions for all other users on the system. Once again, a hyphen means the permission isn't granted.

To provide a practical example, consider the file permissions string for `project_r.txt`: `-rw-rw-r-`. The first character, a hyphen `'-'`, specifies it's a file. The `'r'` characters at the second, fifth, and eighth positions indicate that read permissions are universal—applicable to the user, group, and others. The `'w'` at the third and sixth positions reveals that write permissions are reserved solely for the user and group. Absence of `'x'` confirms that execute permissions are not granted to any category of users.

This level of granularity in permissions is vital for enhancing security, as it allows for fine-tuned control over who can interact with system files and directories and in what manner.

Change file permissions

The company decided that no external users (referred to as `‘other’`) should have the ability to modify, or `‘write’` , to any of its files. When I reviewed the file permissions, I noticed that `project_k.txt` still allowed such write access to `'other'`.

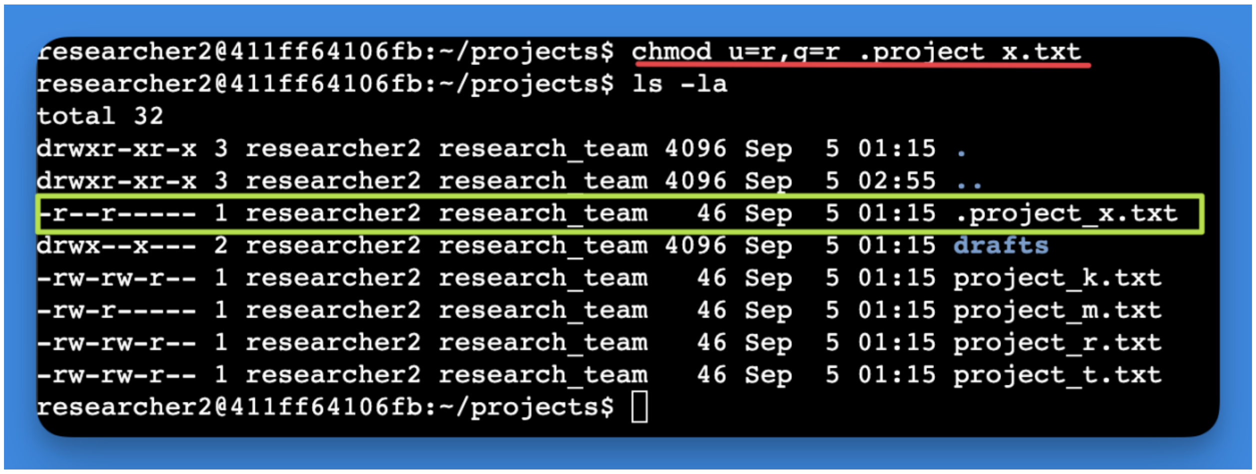


```
researcher2@907bc48642f1:~/projects$ chmod o-w project_k.txt
researcher2@907bc48642f1:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 04:02 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 04:12 ..
-rw--w---- 1 researcher2 research_team   46 Sep  5 04:02 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep  5 04:02 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 04:02 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep  5 04:02 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 04:02 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 04:02 project_t.txt
researcher2@907bc48642f1:~/projects$ 
```

The initial two lines of the displayed snapshot capture the commands I inputted. The subsequent lines showcase the results of the second command. I utilized the `‘chmod’` command, which is designed to modify permissions on files and directories. The first part of the command defines which permissions to alter, while the next part pinpoints the exact file or directory. In this instance, I revoked `write` access for `'other'` on the `project_k.txt` file. To confirm the changes, I ran `ls -la` to get an updated view of the permissions.

Change file permissions on a hidden file

The research team at my organization recently archived `project_x.txt`. They do not want anyone to have write access to this project, but the user and group should have read access.

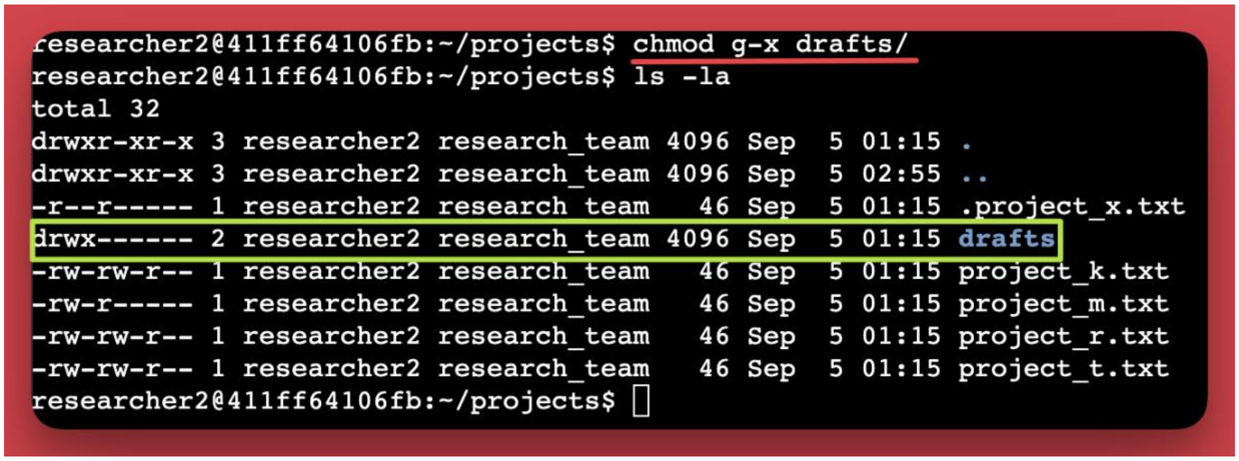


```
researcher2@411ff64106fb:~/projects$ chmod u=r,g=r .project_x.txt
researcher2@411ff64106fb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 01:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 02:55 ..
-r--r----- 1 researcher2 research_team   46 Sep  5 01:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep  5 01:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep  5 01:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_t.txt
researcher2@411ff64106fb:~/projects$ 
```

The screenshot shows the commands I typed in the first two lines, followed by the output of the second command in the remaining lines. I recognized `.project_x.txt` as a hidden file due to its leading period (.). In this case, rather than removing write access for the user and the group while adding read permissions, I used the equal `'='` sign to explicitly set read-only permissions for both. Specifically, I used `u=r,g=r` in the command. This approach effectively wiped out any other permissions that were set, leaving only read permissions active for users and groups.

Change directory permissions

My organization only wants the `researcher2` user to have access to the `drafts` directory and its contents. This means that no one other than researcher2 should have execute permissions.



```
researcher2@411ff64106fb:~/projects$ chmod g-x drafts/
researcher2@411ff64106fb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 01:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep  5 02:55 ..
-r--r----- 1 researcher2 research_team   46 Sep  5 01:15 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Sep  5 01:15 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep  5 01:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep  5 01:15 project_t.txt
researcher2@411ff64106fb:~/projects$ 
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. Earlier, I had found that the `'group'` had execute permissions, which I then removed using the `'chmod'` command. The user `'researcher2'` already possessed execute permissions, so there was no need to add them

Summary

I undertook a series of steps to align the permissions of files and directories within the `'projects'` directory, ensuring they met my organization's specified levels of authorization. I initiated the process with the `ls -la` command to get a comprehensive view of existing permissions. This served as the foundation for the informed adjustments that followed. Utilizing the `chmod` command on several occasions, I successfully modified the permissions for both files and directories to meet the organizational security guidelines. This multi-step approach ensured a more secure and compliant file system within the 'projects' directory