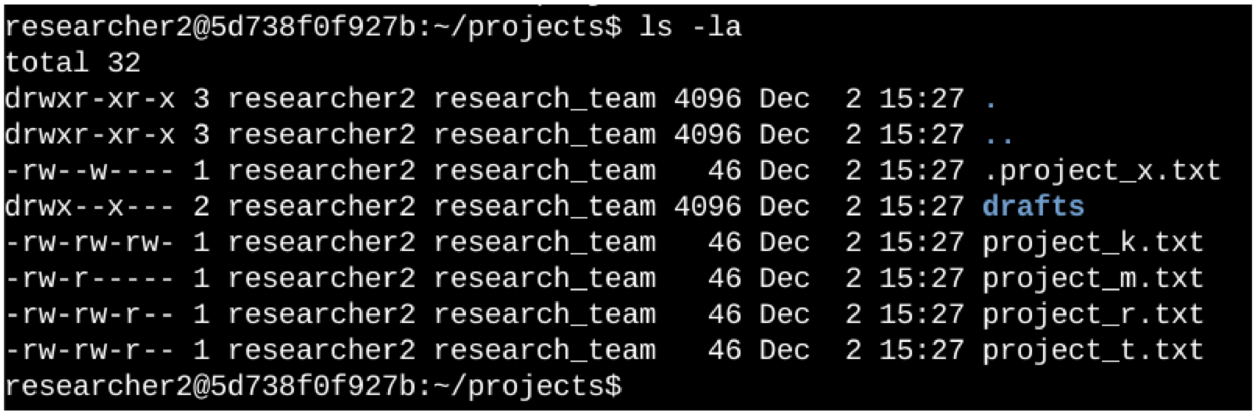
File permissions in Linux

Project description

The research team at my organization requires a comprehensive analysis and update of file permissions within the projects directory to ensure proper authorization levels are established. The current permissions are not aligned with the appropriate access requirements, which poses a security risk to the system. In order to enhance system security, I undertook the following tasks to examine and modify the file permissions:

Check file and directory details

I utilized Linux commands to investigate the existing permissions set for a specific directory in the file system.



```
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-rw--w---- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

The screenshot showcases the executed command and its corresponding output. By employing the ls command with the -la option, I obtained a detailed listing of the contents within the projects directory, including hidden files. The output confirmed the presence of one directory named 'drafts,' one hidden file named '.project_x.txt,' and five other project files. The output also provided valuable information regarding the permissions set for each file or directory. The 10-character string in the first column represents the specific permissions assigned to each entity. These permissions determine the level of access granted to various user groups, such as the owner, group members, and others. By examining the output, I gained insight into the current permission settings within the projects directory. This analysis allowed me to assess whether the permissions aligned with the intended authorization levels for the research team. Identifying any discrepancies between the current permissions and the desired authorization level was crucial for ensuring a secure system environment. By leveraging Linux commands and scrutinizing the permissions of the files and directories within the projects directory, I could proceed with the subsequent steps to rectify any authorization mismatches and reinforce the overall security of the system.

Describe the permissions string

The 10-character string representing file permissions provides valuable insights into the authorized access and specific permissions associated with each file. The breakdown of characters and their respective meanings is as follows:

- 1st Character: The first character, either a 'd' or a hyphen ('-'), denotes the file type. A 'd' indicates a directory, while a hyphen signifies a regular file.

- 2nd-4th Characters: These characters represent the read ('r'), write ('w'), and execute ('x') permissions for the user. A hyphen ('-') in any of these positions indicates the absence of the respective permission for the user.

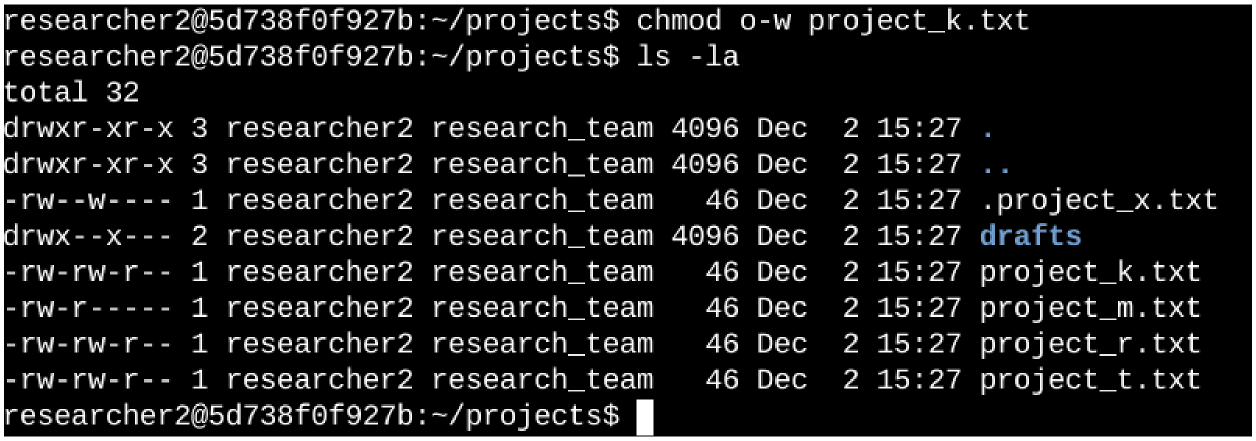
- 5th-7th Characters: These characters indicate the read ('r'), write ('w'), and execute ('x') permissions for the group. Similarly, a hyphen ('-') in any of these positions signifies the lack of the respective permission for the group.

- 8th-10th Characters: These characters represent the read ('r'), write ('w'), and execute ('x') permissions for others, encompassing all users on the system who are neither the owner nor part of the group. Again, a hyphen ('-') in any of these positions indicates the absence of the respective permission for others.

For instance, let's consider the file 'project_t.txt' with the permission string '-rw-rw-r--'. The leading hyphen indicates that 'project_t.txt' is a regular file rather than a directory. The second, fifth, and eighth characters are 'rw,' indicating that the user, group, and others have read permissions. The third and sixth characters are 'rw,' signifying that the user and group possess write permissions. However, the execute permission ('x') is absent for all entities, as indicated by the hyphen in the fourth, seventh, and tenth positions. By understanding the structure and meaning of the permission string, we can interpret and assess the access privileges granted to various users and groups. This comprehension enables us to evaluate whether the current permissions align with the intended authorization levels and take appropriate measures to modify them for optimal system security.

Change file permissions

In alignment with the organization's security requirements, it was determined that 'other' users should not have write access to any files. To ensure compliance, I referred to the previously obtained file permissions and identified 'project_k.txt' as a file that needed write access removed for 'other' users. To achieve this, I utilized the following Linux commands:

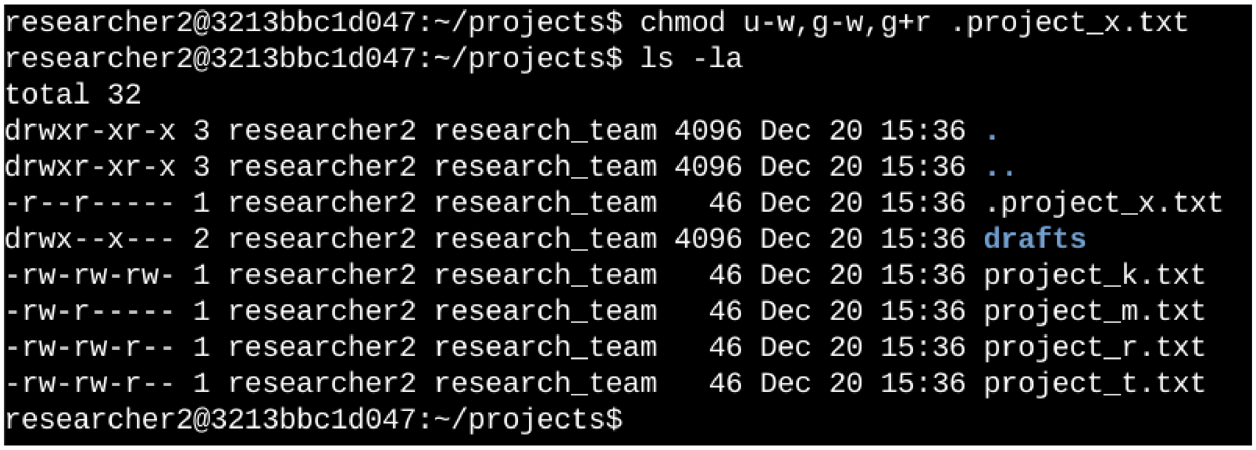


```
researcher2@5d738f0f927b:~/projects$ chmod o-w project_k.txt
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-rw--w---- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

The first two lines of the screenshot showcase the executed commands, while the subsequent lines display the output of the second command. The 'chmod' command serves the purpose of modifying permissions on files and directories. The first argument specifies the permissions to be changed, and the second argument denotes the target file or directory. In this specific example, I employed 'chmod o-w' to remove write permissions for 'other' users from the 'project_k.txt' file. The 'o-w' directive signifies the removal of write permissions specifically for 'other' users. Following this, I executed 'ls -la' to review and confirm the updates I made to the file. By leveraging these commands, I successfully enforced the necessary permission changes to restrict write access for 'other' users on the 'project_k.txt' file. This action helps maintain a secure environment by aligning the file permissions with the organization's policy. The revised output of 'ls -la' after the permission modification accurately reflects the updated permissions, confirming that the write access for 'other' users has been successfully revoked.

Change file permissions on a hidden file

To meet the specific access requirements for the recently archived file 'project_x.txt,' the research team at my organization mandated that write access should be revoked for all users except the owner and group, while the group should have read access. I utilized Linux commands to implement the necessary permission changes, as demonstrated below:



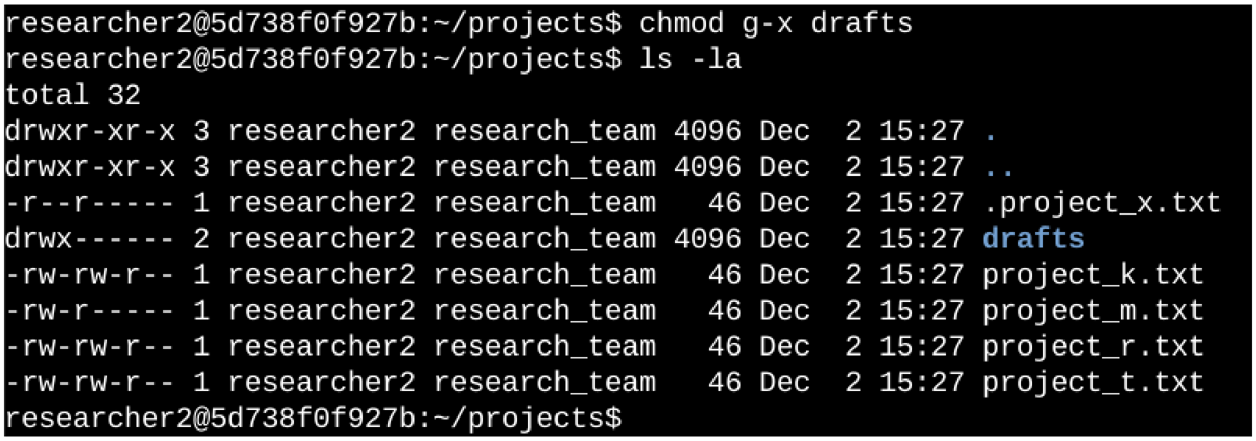
```
researcher2@3213bbc1d047:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@3213bbc1d047:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 ..
-r--r----- 1 researcher2 research_team   46 Dec 20 15:36 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 20 15:36 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec 20 15:36 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 20 15:36 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 20 15:36 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 20 15:36 project_t.txt
researcher2@3213bbc1d047:~/projects$
```

The first two lines of the provided screenshot display the executed commands, while the subsequent lines exhibit the output of the second command. Since '.project_x.txt' begins with a period ('.'), it is recognized as a hidden file in Linux. In this specific example, I employed the 'chmod' command to modify permissions. To begin, I used 'u-w' to remove write permissions from the owner (user). Following this, I utilized 'g-w' to remove write permissions from the associated group. Finally, I employed 'g+r' to add read permissions specifically for the group. By executing these commands, I successfully achieved the desired permission changes for 'project_x.txt.' Write access was effectively removed for both the owner and group, ensuring that only read access remains for these entities. The revised permissions align with the research team's requirements for the archived file. Upon running 'ls -la' after the permission modification, the updated permissions are accurately reflected in the output, thereby validating the successful implementation of the required changes. This action further strengthens the system's security by restricting write access and granting appropriate read access to authorized users.

Change directory permissions

To align with organizational security requirements, it was specified that only the 'researcher2' user should have access to the 'drafts' directory and its contents. Consequently, no other user, group, or entity should possess execute permissions for this directory.

I employed the following Linux commands to achieve the necessary permission changes:



```
researcher2@5d738f0f927b:~/projects$ chmod g-x drafts
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-r--r----- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Dec  2 15:27 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec  2 15:27 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec  2 15:27 project_t.txt
researcher2@5d738f0f927b:~/projects$
```

The first two lines of the provided screenshot display the executed commands, while the subsequent lines exhibit the output of the second command. Through prior investigation, it was established that the group had execute permissions for the 'drafts' directory, which needed to be removed. Using the 'chmod' command, I applied 'g-x' to eliminate execute permissions specifically for the group. As the 'researcher2' user already possessed the necessary execute permissions, no additional steps were required for this user. By executing these commands, I successfully enforced the desired permission changes for the 'drafts' directory. The removal of execute permissions for the group ensures that access is exclusively limited to the 'researcher2' user, in line with the organizational requirements. Running 'ls -ld' after the permission modification validates the updated permissions for the 'drafts' directory, thereby confirming the successful implementation of the changes. This action enhances the overall security of the system by restricting execute permissions and granting exclusive access to the designated user.

Summary

To align the permissions of files and directories within the projects directory with the desired level of authorization established by my organization, I performed a series of actions. These actions were aimed at reviewing and modifying permissions as necessary. The summary of my approach is as follows:

1. Initial Permissions Check: To begin, I used the ls -la command to comprehensively examine the existing permissions for the projects directory. This detailed listing provided crucial insights into the current access privileges, serving as a foundation for subsequent decision-making.

2. Informed Decision-Making: The information obtained from the permissions check played a pivotal role in shaping my subsequent actions. By carefully analyzing the permissions, I identified areas where modifications were required to ensure proper authorization levels.

3. Multiple Permission Modifications: Utilizing the chmod command, I made multiple iterations of permission changes to align with the desired authorization levels. This involved selectively adjusting permissions on both files and directories within the projects directory. The specific modifications were determined based on the requirements outlined by my organization.

By employing these steps, I effectively reviewed and adjusted the permissions within the projects directory to match the desired level of authorization. This proactive measure enhances system security by ensuring that access privileges are appropriately granted and unauthorized access is restricted. The combined use of the ls -la command for initial assessment and the chmod command for making multiple permission modifications facilitated the process of aligning the permissions with the established authorization requirements. By following this approach, I successfully modified the permissions to match the desired level of authorization within the projects directory, thereby bolstering the overall security of the system.