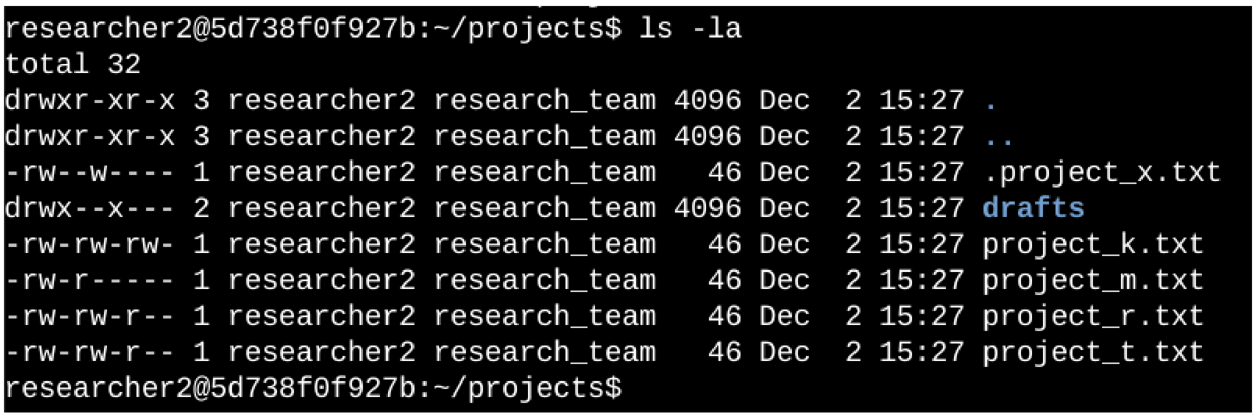
File permissions in Linux

Project description

The research team at my organization must revise the file permissions for specific files and directories within the projects directory. The current permissions do not align with the required level of authorization. Ensuring these permissions are checked and updated will enhance system security. To accomplish this, I undertook the following actions:

Check file and directory details

Here is an example of how I utilized Linux commands to ascertain the current permissions configured for a particular 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$
```

In the screenshot, the initial line shows the command I inputted, while the subsequent lines show the resulting output. The command lists all items within the projects directory. I employed the `ls` command with the `-la` option to generate a comprehensive listing of file contents, including hidden files. The output reveals one directory named "drafts," one concealed file named ".project_x.txt," and five additional project files. The 10-character string in the first column indicates the permissions assigned to each file or directory.

Describe the permissions string

The 10-character string can be analyzed to determine access permissions for each file or directory. Here's how each character represents specific permissions:

- **1st character:** Indicates the file type. If it's a "d," it denotes a directory. If it's a hyphen "-", it signifies a regular file.

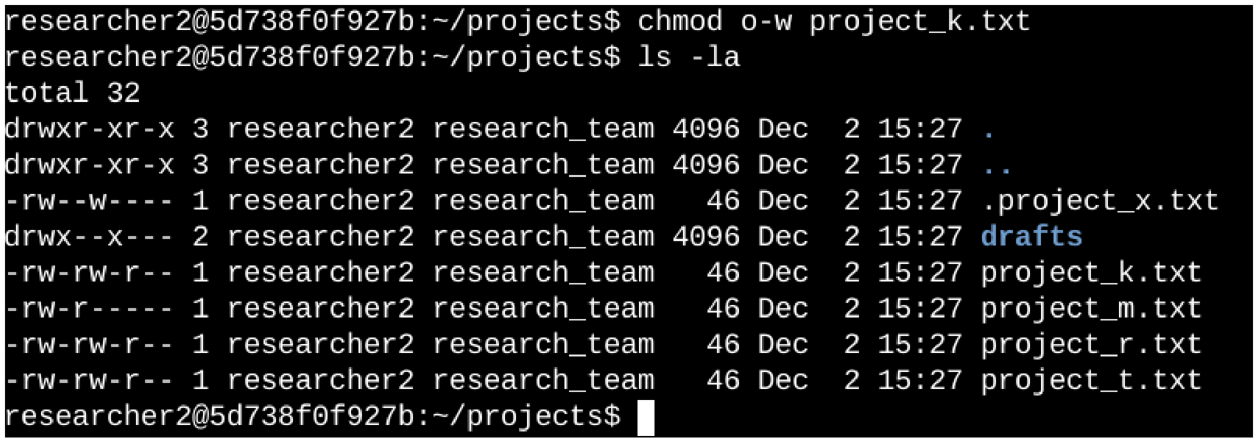
- **2nd-4th characters:** Represent permissions for the user (owner) in the order of read (r), write (w), and execute (x). A hyphen "-" indicates the absence of that permission.
- **5th-7th characters:** Indicate permissions for the group in the same sequence (read, write, execute).
- **8th-10th characters:** Show permissions for others (all users not falling into the user or group categories) in the same sequence (read, write, execute).

For example, consider the file permissions for "project_t.txt" as "-rw-rw-r--". The first character "-" indicates it's a file. The user has read and write permissions (rw), as do the group and others (rw-). No execute permissions are granted to any category.

Change file permissions

The organization decided to revoke write access for "other" on all their files. To adhere to this policy, I reviewed the file permissions I had previously retrieved. I identified that "project_k.txt" needed to have write access removed for "other."

Here's an example of how I executed this using Linux commands:



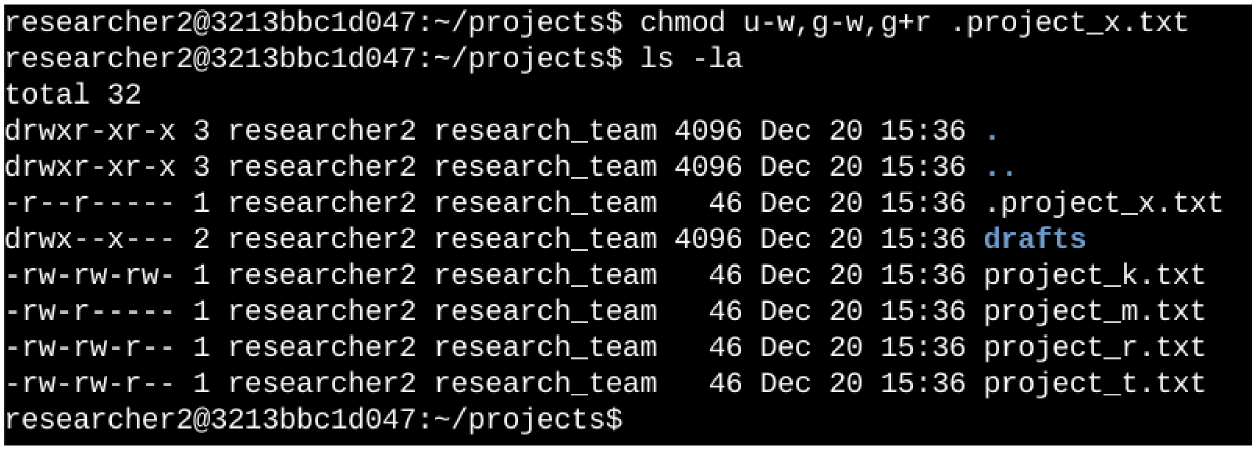
```
researcher2@5d738f0f927b:~/projects$ chmod o-w project_k.txt
researcher2@5d738f0f927b:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec  2 15:27 ..
-rw--w---- 1 researcher2 research_team   46 Dec  2 15:27 .project_x.txt
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 initial two lines in the screenshot show the commands I inputted, with subsequent lines displaying the output of the second command. The `chmod` command is employed to modify permissions on files and directories. The first parameter specifies which permissions are to be altered, while the second specifies the file or directory affected. In this instance, I revoked write permissions for "other" on the file "project_k.txt". Following this adjustment, I utilized `ls -la` to inspect the changes I implemented.

Change file permissions on a hidden file

The research team at my organization has recently archived "project_x.txt". They have specified that no one should have write access to this project, while the user and group should retain read access.

Here's an example demonstrating how I utilized Linux commands to adjust these permissions:



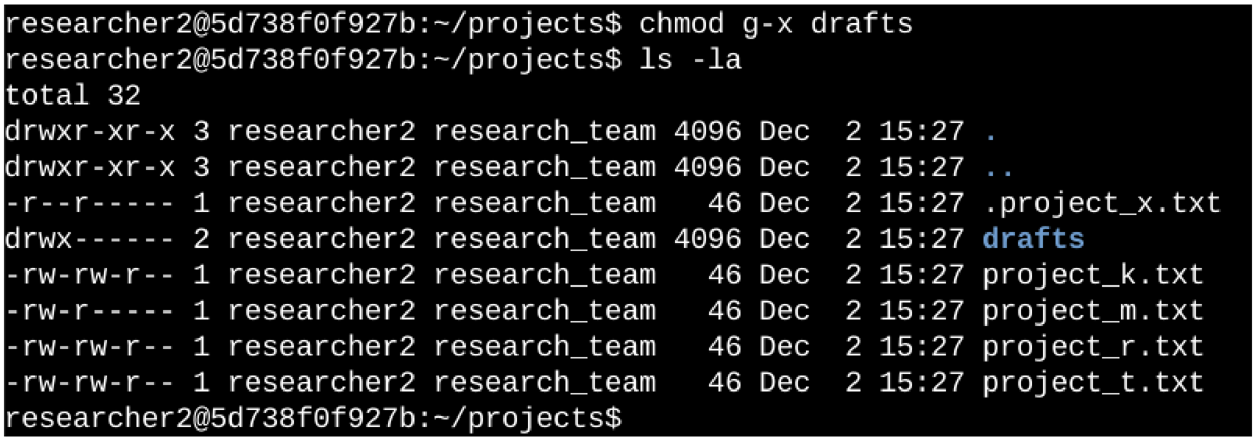
```
researcher2@3213bbc1d047:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@3213bbc1d047:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 20 15:36 ..
-r--r----- 1 researcher2 research_team   46 Dec 20 15:36 .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 initial two lines in the screenshot show the commands I inputted, with subsequent lines displaying the output of the second command. I identified ".project_x.txt" as a hidden file because its name begins with a period (.). In this scenario, I adjusted permissions by removing write access for both the user and group, and additionally granted read access to the group. Specifically, I revoked write permissions for the user using "u-w", followed by removing write permissions for the group with "g-w", and finally, I added read permissions for the group using "g+r".

Change directory permissions

My organization has restricted access to the drafts directory and its contents solely to the user "researcher2". As a result, only researcher2 should have execute permissions.

Here's an example demonstrating how I applied Linux commands to adjust these permissions:



```
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 displayed output lists permissions for multiple files and directories. Line 1 denotes the current directory ("projects"), and line 2 represents the parent directory ("home"). Line 3 shows a regular file named ".project_x.txt". Line 4 displays the "drafts" directory with restricted permissions, where only researcher2 has execute access. Previously, it was decided that the group had execute permissions, which I removed using the chmod command. Since researcher2 already had execute permissions, no further action was required for them.

Summary

I adjusted various permissions in accordance with my organization's authorization requirements for files and directories within the projects directory. Initially, I used `ls -la` to review the current permissions, which guided my subsequent actions. I proceeded by utilizing the `chmod` command multiple times to modify permissions on specific files and directories.