File permissions in Linux

Project description:

For this project the research team at my organization was tasked with updating the file permissions for specific files and directories within the `projects` directory. Currently, these permissions do not align with the appropriate level of authorization. By checking and updating these permissions we can enhance the security of our system. To accomplish this goal, I undertook the following actions:

Check file and directory details:

The code below shows how I used Linux commands to find out the current 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$
```

In the screenshot, the first line shows the command I inputted, and the following lines display the output. The code shows all items within the `projects` directory. I used the `ls` command and the `-la` option to reveal a comprehensive listing of file contents, including hidden files. The output of my command denotes the existence of a single directory labeled `drafts`, one hidden file named `.project_x.txt`, and five additional project files. The string of 10 characters in the first column represents the permissions assigned to each file or directory.

Describe the permissions string:

The 10-character string can be analyzed to identify who has permission to access the file and the exact permissions granted. The characters and what they represent are as follows:

- **1st character**: This character of the string can be either a `d` or hyphen (`-`) and signifies the file type. If it's a `d`, it signifies a directory. If it's a hyphen (`-`), it signifies a regular file.
- **2nd-4th characters**: The read (`r`), write (`w`), and execute (`x`) characters represent 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 (`-`) it shows that the user has not been granted permission.
- **8th-10th characters:** The read (`r`), write (`w`), and execute (`x`) characters signify the permissions for other. This owner type encompasses all other users on the system apart from the user and the group. If one of these characters is a hyphen (`-`) instead, it means that the permission is not granted for other.

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

Change file permissions:

The organization decided that other should not have write access to any of their files. To adhere to this policy, I reviewed the file permissions that I had previously returned. Upon review, I found that `project_k.txt` needed to have write access removed for other.

Below is the code I used to accomplish 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$
```

In the screenshot, the first two lines show the commands I inputted, while the subsequent lines show the output of the second command. The `chmod` command is utilized to modify the permissions on files and directories. The first argument shows what permissions should be changed, and the second argument specifies the file or directory. In this example, I revoked write permissions from other for the `project_k.txt` file. Following this, I utilized `ls -la` to review the changed I implemented.

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 will have read access.

Below is the code I used to accomplish this using Linux commands to change the 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 first two lines of the screenshot show the commands I entered, and the other lines reflect the output of the second command. I know that `.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

My organization only wants the `researcher2` user to have access to the `drafts` directory and its contents. This will mean that no one other than `researcher2` should have execute permissions. Further explanation listed on the next page.

Below is the code I used to accomplish this using Linux commands to change the 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$
```

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

Summary

I adjusted various permissions to align with the authorization standards required by my organization for files and directories within the `projects` directory. The first step in this was using `ls -la` to check the permissions for the directory. This informed my decisions in the following steps. I then used the `chmod` command multiple times to modify permissions on both files and directories accordingly. Ensuring that permissions are appropriately set is crucial for maintaining the security and integrity of our organization's data. By adjusting permissions to match our standards, we reinforce our systems resilience against unauthorized access and potential security breaches.