File permissions in Linux

Project description

The research team at my organization needs to update the file permissions for certain files and directories within the projects directory. The permissions do not currently reflect the level of authorization that should be given. Checking and updating these permissions will help keep their system is secure. To complete this task, I performed the following tasks:

Check file and directory details

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

```
researcher2@3144276483d6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:20 .
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:52 ..
-rw--w---- 1 researcher2 research_team   46 May 12 07:20 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 May 12 07:20 drafts
-rw-rw-rw- 1 researcher2 research_team   46 May 12 07:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 May 12 07:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_t.txt
researcher2@3144276483d6:~/projects$
```

- 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 `drwxr-xr-x` in the first column represents the permissions set on each file or directory.

Describe the permissions string

The 10-character string `drwxr-xr-x` 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 organization determined that other shouldn't have write access to any of their files. To comply with this, I referred to the file permissions that I previously returned. I determined `project_k.txt` must have the write access removed for other.

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

```
researcher2@3144276483d6:~/projects$ chmod o-w project_k.txt
researcher2@3144276483d6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:20 .
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:52 ..
-rw--w---- 1 researcher2 research_team   46 May 12 07:20 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 May 12 07:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 May 12 07:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_t.txt
researcher2@3144276483d6:~/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.
- 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

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

```
researcher2@3144276483d6:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@3144276483d6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:20 .
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:52 ..
-r--r----- 1 researcher2 research_team   46 May 12 07:20 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 May 12 07:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 May 12 07:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_t.txt
researcher2@3144276483d6:~/projects$
```

- The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command.
- `.project_x.txt` is a hidden file because it starts with a period (`.`). In the screenshot, 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

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

```
researcher2@3144276483d6:~/projects$ chmod g-x drafts
researcher2@3144276483d6:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:20 .
drwxr-xr-x 3 researcher2 research_team 4096 May 12 07:52 ..
-r--r----- 1 researcher2 research_team   46 May 12 07:20 .project_x.txt
drwx------ 2 researcher2 research_team 4096 May 12 07:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 May 12 07:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 May 12 07:20 project_t.txt
researcher2@3144276483d6:~/projects$
```

- The output displays the permission listing for several files and directories.
- Line 1 indicates the current directory (projects), and line 2 indicates the parent directory (home).
- Line 3 indicates a regular file titled `.project_x.txt`. Line 4 is the directory `drafts` with restricted permissions. Here you can see that only researcher2 has execute permissions. It was previously determined that the group had execute permissions,
- 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 project focused on managing file and directory permissions within the `projects` directory to align with organizational access policies. Initial permissions were reviewed using `ls -la`, followed by multiple `chmod` operations to enforce the required permission levels across files and subdirectories.