File Permissions in Linux

Project description

I am a security analyst for an organization, currently working with the research team. Today, I am tasked with looking over the current permissions, to make sure the users on the research team are authorized with the appropriate permissions.

Check file and directory details

```
researcher2@e001936e485a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:47 ..
-rw--w---- 1 researcher2 research_team   46 Sep 25 06:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 25 06:08 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 25 06:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 25 06:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_t.txt
```

- After navigating to the `projects` directory, we input the command `ls -la` to display the file and directory permissions (this command also includes hidden files and directories)

Describe the permissions string

```
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_t.txt
```

- Observing `project_t.txt`, the permissions of this file are set to `-rw-rw-r--`
  - The first character is “-” and signifies that this is a file. If the first character were “d” then this would signify that this is a directory
  - The next three characters, which represent the “user”, is “rw-”. This means that the user has read (r) and write (w) permissions, but not execute (x) permissions. No permission is signified by “-”.
  - The second set of three characters, which represent the “group”, is “rw-”. So, the group has the same permissions as the user.
  - The third set of three characters, which represent “others”, is “r--”. The only permission other has is the read permission. Other does not have write or execute permissions.

Change file permissions

The organization does not allow other to have the write permission for any files. Looking at the files, we see that `project_k.txt` has write permissions for other. So we must change its file permissions. We perform the following command:

```
researcher2@e001936e485a:~/projects$ chmod o-w project_k.txt
```

- `chmod` is a command that adds or removes permissions. I used this command to remove the write permission from other for `project_k.txt`

We can now check to make sure that the write permission has been successfully removed from other:

```
researcher2@e001936e485a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:47 ..
-rw--w---- 1 researcher2 research_team   46 Sep 25 06:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 25 06:08 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 25 06:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_t.txt
```

Change file permissions on a hidden file

The research team has archived `.project_x.txt`, which is why it's a hidden file (“.” at the beginning of a file name indicates it’s a hidden file). There should be no write permissions on this file, but the user and group should have read permissions. First, we must add read permission for the group, then remove write permission for the group, and lastly remove write permission for the user:

```
researcher2@e001936e485a:~/projects$ chmod g+r .project_x.txt
researcher2@e001936e485a:~/projects$ chmod g-w .project_x.txt
researcher2@e001936e485a:~/projects$ chmod u-w .project_x.txt
```

And then we can check to make sure the new permissions were successfully implemented:

```
researcher2@e001936e485a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:47 ..
-r--r----- 1 researcher2 research_team   46 Sep 25 06:08 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 25 06:08 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 25 06:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_t.txt
```

Change directory permissions

The files and subdirectories in the `projects` directory are owned by researcher2, so only researcher2 should have access to the `drafts` directory, but currently the group has execute permission. To remove the permission we input the following command:

```
researcher2@e001936e485a:~/projects$ chmod g-x drafts
```

Now, we can verify that researcher2 has the only access to this directory:

```
researcher2@e001936e485a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:08 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 25 06:47 ..
-r--r----- 1 researcher2 research_team   46 Sep 25 06:08 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Sep 25 06:08 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 25 06:08 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 25 06:08 project_t.txt
```

Summary

After looking over the permissions, I noted that some permissions did not match the authorization that should be given. So I corrected the unauthorized permissions, so that user access would align with the organization's security standards.