File permissions in Linux

Project description

As part of my responsibilities supporting the research team at my organization, I was tasked with updating file and directory permissions within the `projects` directory. Some existing permissions did not align with the intended access levels, posing potential security risks. To improve access control and secure sensitive data, I reviewed and modified the file system permissions accordingly.

Check file and directory details

To begin, I examined the existing permissions for all the files and directories using Linux commands.

`ls -la projects`

```
researcher2@e06c66fc3968:~$ cd projects/
researcher2@e06c66fc3968:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:28 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:58 ..
-rw--w---- 1 researcher2 research_team   46 Jul 10 17:28 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 10 17:28 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jul 10 17:28 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 10 17:28 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_t.txt
```

This command provided a detailed listing of all files, including hidden ones, within the `projects` directory. The output showed one subdirectory named drafts, a hidden file `.project_x.txt`, and several project files. The first column of the output displayed a 10-character permission string for each item.

Describe the permissions string

The permission string helps identify the file type and the access rights assigned to users, groups, and others:

- **1st character**: File type (`d` for directory, `-` for file)
- **2nd–4th**: User permissions (read `r`, write `w`, execute `x`)
- **5th–7th**: Group permissions (read `r`, write `w`, execute `x`)
- **8th–10th**: Other users' permissions (read `r`, write `w`, execute `x`)

For instance, `-rw-rw-r--` on `project_r.txt` indicates it's a regular file where the user and group have read/write access, and others have read-only access.

Change file permissions

To comply with security guidelines, I removed write access for `others` on certain files. For example, `project_k.txt` had inappropriate write permissions for others.

`chmod o-w project_k.txt`
`ls -la`

```
researcher2@e06c66fc3968:~/projects$ chmod o-w project_k.txt
researcher2@e06c66fc3968:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:28 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:58 ..
-rw--w---- 1 researcher2 research_team   46 Jul 10 17:28 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 10 17:28 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 10 17:28 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_t.txt
```

The initial two lines in the screenshot show the commands I typed, while the remaining lines show the result of the second command. The `chmod` command is used to modify file and directory permissions. The first parameter defines which permissions to adjust, and the second identifies the file or directory affected. In this case, I took away write permissions for others on the `project_k.txt` file. I then ran `ls -la` to verify the changes.

Change file permissions on a hidden file

The file `.project_x.txt` had been archived, and the team required only read access for the user and group, with no write permissions.

`chmod u-w,g-w,g+r .project_x.txt`
`ls -la`

```
researcher2@e06c66fc3968:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@e06c66fc3968:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:28 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:58 ..
-r--r----- 1 researcher2 research_team   46 Jul 10 17:28 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 10 17:28 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 10 17:28 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_t.txt
```

These commands removed write permissions from both the user and the group, and granted read access to the group, effectively limiting modification privileges.

Change directory permissions

Only the `researcher2` user should have access to the drafts directory. This required removing execute permissions from other users.

```
chmod g-x drafts
ls -la
```

```
researcher2@e06c66fc3968:~/projects$ chmod g-x drafts
researcher2@e06c66fc3968:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:28 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 17:58 ..
-r--r----- 1 researcher2 research_team   46 Jul 10 17:28 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Jul 10 17:28 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul 10 17:28 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 17:28 project_t.txt
```

After executing the command, the directory showed that only `researcher2` retained execute permissions, preventing unauthorized access.

Summary

Through careful analysis and modification of permissions in the `projects` directory, I aligned file access with organizational policies. Using tools like `ls -la` and `chmod`, I ensured only authorized users retained appropriate access, reducing the risk of data exposure or misuse.