File permissions in Linux

Project description

I reviewed and updated file system permissions for the research team to ensure access was granted only to authorized users. I identified misconfigurations where "others" or incorrect groups had write or execute access. I corrected these permissions using Linux file system commands to comply with internal security policies.

Check file and directory details

The following screenshot shows how I used Linux commands to check permissions for the specific directory here it is named as projects

```
researcher2@272e5fa93318:~$ cd projects
researcher2@272e5fa93318:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug  6 09:20 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Aug  6 09:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug  6 09:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_t.txt
researcher2@272e5fa93318:~/projects$
```

The first line of the screenshot shows the command `cd projects` followed by `ls -l`, which lists detailed information about files and directories.

The output shows one directory (`drafts`) and four files (`project_k.txt`, `project_m.txt`, `project_r.txt`, `project_t.txt`). Each line starts with a 10-character string that shows the file type and permissions.

- The first character indicates if it's a file (`-`) or directory (`d`).
- The next three sets of characters show read (`r`), write (`w`), and execute (`x`) permissions for the **user**, **group**, and **others**.

Example:

- `drwx--x---` means `drafts` is a directory with full access for the user, execute-only for the group, and no access for others.
- `-rw-rw-rw-` for `project_k.txt` shows read/write access for all, which may be insecure.

Describe the permissions string

The 10-character string can be deconstructed to determine who is authorized to access each file or directory and what permissions they have. The characters represent:

- **1st character**: d for a directory, - for a regular file.
- **2nd–4th characters**: Permissions for the **user** (owner): read (r), write (w), execute (x).
- **5th–7th characters**: Permissions for the **group**.
- **8th–10th characters**: Permissions for **others** (everyone else).

1. **drafts** ⟶ drwx--x---
    - d: It is a directory.
    - rwx: User (researcher2) can read, write, and enter the directory.
    - --x: Group can only execute (can enter but not list contents).
    - ---: Others have no access.
2. **project_k.txt** ⟶ -rw-rw-rw-
    - -: Regular file.
    - rw-: User can read and write.
    - rw-: Group can read and write.
    - rw-: Others can read and write.

    This is a security risk — "others" should not have write access.

3. **project_m.txt** ⟶ -rw-r-----
    - -: Regular file.
    - rw-: User can read and write.
    - r--: Group can only read.
    - ---: Others have no access.

    Secure and restricted.

4. **project_r.txt** ⟶ -rw-r--r--
    - -: Regular file.
    - rw-: User can read and write.
    - r--: Group can read.
    - r--: Others can read.

    Others may not need read access.

5. **project_t.txt** ⟶ -rw-rw-r--
    - -: Regular file.
    - rw-: User can read and write.
    - rw-: Group can read and write.
    - r--: Others can only read.

Read access for others may not be appropriate for confidential data.

Change file permissions

```
researcher2@272e5fa93318:~/projects$ chmod g-r project_m.txt
researcher2@272e5fa93318:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug  6 09:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_k.txt
-rw------- 1 researcher2 research_team   46 Aug  6 09:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_t.txt
```

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 is used to change file permissions in Linux. The first argument (g-r) tells the system to remove read permissions from the **group**, and the second argument (project_m.txt) specifies the file to modify.

In this example, I removed **read** permissions from the **group** for the project_m.txt file. After making this change, I used ls -l to verify the update. The new permission string for project_m.txt is now -rw-------, indicating that only the user has read and write access, and both group and others have no permissions.

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 should have read access.

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

```
researcher2@272e5fa93318:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@272e5fa93318:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug  6 09:20 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug  6 10:34 ..
-r--r----- 1 researcher2 research_team   46 Aug  6 09:20 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug  6 09:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_k.txt
-rw------- 1 researcher2 research_team   46 Aug  6 09:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_t.txt
```

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

In this example:

- I removed **write** permissions from the user with `u-w`.
- I removed **write** permissions from the group with `g-w`.
- I added **read** permissions to the group with `g+r`.

After running `chmod u-w,g-w,g+r .project_x.txt`, I used `ls -la` to confirm the updated permissions. The new permission string for `.project_x.txt` is now `-r--r-----`, which means only the user and group can **read**, and no one has **write** or **execute** access.

Change directory permissions

```
researcher2@272e5fa93318:~/projects$ chmod g-x drafts
researcher2@272e5fa93318:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Aug  6 09:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_k.txt
-rw------- 1 researcher2 research_team   46 Aug  6 09:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug  6 09:20 project_t.txt
```

My organization only wants the `researcher2` user to have access to the `drafts` directory and its contents. This means that no one other than `researcher2` should have execute permissions.

The following code demonstrates how I used Linux commands to change the permissions:
The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I previously determined that the group had execute permissions on the `drafts` directory, so I used the `chmod` command with `g-x` to remove execute access for the group.

Since the `researcher2` user already had execute permissions, no additional changes were needed for the user. After the change, I ran `ls -l` to confirm the update. The permission string for the `drafts` directory is now `drwx------`, meaning only the user has full access, and both group and others have no permissions.

Summary

I changed multiple permissions to match the level of authorization my organization wanted for files and directories in the `projects` directory. The first step in this process was using the `ls -la` command to check the current permissions for all files, including hidden ones. This helped me identify files and directories that had incorrect or overly permissive access.

Based on this review, I used the `chmod` command multiple times to adjust permissions:

- I removed read access from the group for `project_m.txt`.

- I removed write permissions from the user and group, and added read access for the group on the hidden file `.project_x.txt`.
- I also removed execute access from the group for the `drafts` directory to ensure that only the `researcher2` user could access it.

These changes were made to ensure that only authorized users had the appropriate access, while unauthorized access was removed, improving the security of the research team's file system.