File permissions in Linux

Project description

This project demonstrates practical Linux file permission management skills. As a security professional, my task was to review and modify file and directory permissions within a research team's file system. Through precise use of chmod and ls -la commands, I ensured that unauthorized access was removed and sensitive files adhered to strict security protocols, enhancing system security.

Check file and directory details

To begin, I gained a comprehensive understanding of the current file and directory permissions within the projects directory. This involved displaying all files, including hidden ones, and their associated permission strings.

ls -la

```
drwxr-xr-x 3 researcher2 research_team 4096 Jun  8 17:03 .
drwxr-xr-x 3 researcher2 research_team 4096 Jun  8 17:22 ..
-rw--w---- 1 researcher2 research_team   46 Jun  8 17:03 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jun  8 17:03 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jun  8 17:03 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jun  8 17:03 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun  8 17:03 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jun  8 17:03 project_t.txt
```

Describe the permissions string

The **10-character string** at the beginning of each line in the ls -la output represents the file or directory type and its permissions. This string is crucial for understanding who can **read**, **write**, or **execute** a given file or directory.

Using project_k.txt's initial permissions (-rw-rw-rw-) as an example:

- **1st character (-):** Indicates the **file type**. A hyphen (-) means it's a **regular file**. If it were a directory, it would be d.

- **Next 3 characters (rw-):** Represent the permissions for the **User** (owner) of the file.
    - r: Read permission (owner can view contents).
    - w: Write permission (owner can modify or delete).

    - -: Permission not granted (in this case, no execute).

- **Next 3 characters (rw-):** Represent the permissions for the **Group** that owns the file.
    - r: Read permission (users in the file's group can view contents).
    - w: Write permission (users in the file's group can modify or delete).
    - -: Permission not granted (no execute).

- **Last 3 characters (rw-):** Represent the permissions for **Others** (anyone else on the system not in the owner or group categories).
    - r: Read permission (other users can view contents).
    - w: Write permission (other users can modify or delete).
    - -: Permission not granted (no execute).

Change file permissions

Scenario: The organization does not allow others to have write access to any files.

Identification: Based on the initial ls -la output, project_k.txt had permissions -rw-rw-rw-. The "other" category (rw-) had **write access** (w), which was a security vulnerability.

Linux Command Used to Modify:

chmod o-w project_k.txt

Explanation: The chmod command is used to **change file permissions**.

- o: Specifies that the change applies to **"other" users**.
- -: Indicates that a permission is being **removed**.
- w: Specifies the **"write" permission**. This command successfully removed write access for all users categorized as "other" on the project_k.txt file.

Output (After modification and ls -la):

-rw-rw-r-- 1 researcher2 research_team 46 Jun 8 17:03 project_k.txt

(Note the change from rw- to r-- in the third permission triplet for project_k.txt)

Change file permissions on a hidden file

Scenario: The hidden file .project_x.txt should not have write permissions for anyone, but the user and group should be able to read the file.

Identification: Initially, .project_x.txt had permissions -rw--w----. Both the **user** (rw-) and the **group** (-w-) incorrectly had write permissions according to the new policy. The group also lacked read permission, which was required.

Linux Command Used to Modify:

chmod u=r,g=r .project_x.txt

Explanation: This chmod command uses the **= operator** to explicitly **set** the permissions for the user and group.

- u=r: Sets the **user's** permissions to **"read only"** (r--).
- g=r: Sets the **group's** permissions to **"read only"** (r--).
- The absence of o= means that "other" permissions remain as they were (which was ---), satisfying the "no write for anyone" rule. This command ensured that only the owner and members of the group could read the .project_x.txt file, and no one could write to it.

Output (After modification and ls -la):

-r--r----- 1 researcher2 research_team 46 Jun 8 17:03 .project_x.txt

(Note the change from rw--w---- to r--r----- for .project_x.txt)

Change directory permissions

Scenario: The files and directories in the projects directory belong to the researcher2 user. Only researcher2 should be allowed to access the drafts directory and its contents.

Identification: The drafts directory initially had permissions drwx--x---. This meant the group had execute permission, allowing some level of access. To ensure **only the owner** (researcher2) has full access (read, write, execute), permissions should be drwx------.

Linux Command Used to Modify:

chmod 700 drafts

Explanation: This chmod command uses the **octal (numeric) mode** to set permissions precisely.

- 7: Represents rwx (read, write, execute) for the **owner**.
- 0: Represents --- (no permissions) for the **group**.
- 0: Represents --- (no permissions) for **others**. This command effectively locked down the drafts directory, granting full control exclusively to its owner, researcher2.

Output (After modification and ls -la):

drwx------ 2 researcher2 research_team 4096 Jun 8 17:03 drafts

(Note the change from drwx--x--- to drwx------ for drafts)

Summary

In this activity, I identified and corrected several permission misconfigurations within a Linux file system. I successfully removed unauthorized write access from a critical file (project_k.txt) and secured a hidden archive (.project_x.txt), ensuring only specific users could read it while preventing any write access. Finally, I locked down the drafts directory, granting full control solely to its owner. These actions were crucial for reinforcing the organization's security posture by ensuring users had only the necessary authorization.