File permissions in Linux

Project description

Within my organization, the research team identified a need to update the permissions assigned to certain files and directories within the projects directory. The existing permissions were not in sync with the level of authorization that should have been granted, creating a potential security vulnerability.
To bolster the security of their system and ensure that only authorized individuals had access to sensitive information, I took on the task of reviewing and adjusting these permissions. In essence, this process involved controlling who could read, write, or execute specific files and folders.
By aligning permissions with the appropriate levels of authorization, we aimed to enhance the system's security posture. It's somewhat akin to making sure that the doors in a building have locks that match the level of access needed - no more, no less. This helps prevent unauthorized access, mitigating potential risks and ensuring that sensitive data remains protected.
To accomplish this important task, I carried out a series of actions and checks to meticulously examine and update permissions, ensuring that the research team's system maintained its integrity and security standards.

Check file and directory details

At the directory /home/researcher2, I navigated to the projects directory by using the **cd** command. I used the ls -la command to list the content and permissions. The permissions of the files in the `projects` directory are as follows:

```
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 14 18:40 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 14 18:40 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 14 18:40 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 14 18:40 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 14 18:40 project_t.txt
```

Describe the permissions string

Reading permissions:

Example & Meaning	**Example & Meaning**
D = Directory	**X** = execute
- = file	**U** = User, user always comes first on the permission.
R = read	**G** = Group, always comes second.
W = write	**O** = Other, always comes last.
Example of directory permissions	**Example & Meaning**
d**rwx**rwxrwx = rwx is the permission of the user	If the user, group or other lack permission to something. You'll see a **-**
drwx**rwx**rwx = the second part is the group permission.	• Using **+ adds permissions to the user, group, or other** • Using - **removes permissions from the user, group, or other** • Using **= assigns permissions for the user, group, or other**
Drwxrwx**rwx =** the last part is the Other permission	

Change file permissions

- You need to set in place the **principle of least privilege,** which is the concept of granting only the minimal access and authorization required to complete a task or function.
- You can use the **`chmod`** command to change permissions on files and directories.
- The **`chmod`** command requires two arguments. The first argument indicates how to change permissions, and the second argument indicates the file or directory that you want to change permissions for. For example, the following command would add all permissions to **`login_sessions.txt`**:
- **`chmod u+rwx,g+rwx,o+rwx login_sessions.txt`**

Change file permissions on a hidden file

First you need to know what a hidden file looks like.

- Hidden files start with a period (.) at the beginning.
- You can use ls -a command to display hidden files

  For example: look how the green colored file starts with a (.), that's our hidden file
  -rw--w---- 1 researcher2 research_team 46 Sep 28 08:37 .project_x.txt
  drwx--x--- 2 researcher2 research_team 4096 Sep 28 08:37 drafts
  Using the chmod command I changed the permissions to this file.
  Look at the red text. Both the user and the group now have only read permissions.

  -r--r----- 1 researcher2 research_team 46 Sep 28 08:37 .project_x.txt

Change directory permissions

This is what the directory permissions look like before we change the permissions.

drwx--x--- 2 researcher2 research_team 4096 Sep 28 08:37 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 28 08:37 ..
We are gonna use the chmod command to change the permissions

chmod g-x drafts this is what the command will look like.

After the change of directory permissions:
drwx------ 2 researcher2 research_team 4096 Sep 28 08:37 .
drwxr--- r-x 3 researcher2 research_team 4096 Sep 28 08:37 ..

Summary

I took responsibility for managing directory and file permissions, a crucial task for a security analyst. This involves controlling who can access and manipulate various files and directories on a computer system.
To perform this task effectively, I utilized the **'ls'** command with specific options like **'-l'** and **'-la'**. These options allowed me to thoroughly examine and investigate the permissions associated with directories and files. Essentially, it's like peering into a detailed map of who has the keys to access specific doors and what they can do once inside.
In addition to examining permissions, I also utilized the **'chmod'** command. This command gave me the power to change and adjust user permissions. Here's where the principle of least privilege comes into play. The principle of least privilege is like a security guideline that suggests granting users only the minimum level of access or permissions they need to perform their tasks. It's like ensuring that everyone in a building has just enough access to the rooms they require and nothing more.
By using '**chmod**', I could fine-tune these permissions, aligning them with the principle of least privilege. This approach enhances system security by limiting the potential for misuse or unauthorized access. In essence, it's a way of tightening security bolts and making sure that only those with a legitimate need can access and manipulate sensitive data and files.