File permissions in Linux

Project description

This project will take a look into a hypothetical organization's file system and delve into the `projects` subdirectory. The file permissions are not configured correctly as they grant unnecessary authorization to different users and groups, posing security risks and vulnerabilities. I will check and update these permissions to ensure the system is secured. I will also look over file types and user permissions for each file in the `projects` subdirectory and modify user, group, and other permissions to implement the principle of least privilege. To achieve this security hardening task, I'll utilize the `ls` and `chmod` command-line utilities.

Check file and directory details

I used the `ls -l` linux command to list all files within the `projects` subdirectory in long format:

```
researcher2@5d9b44662eaf:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Jan 21 06:06 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan 21 06:06 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 21 06:06 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 21 06:06 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 21 06:06 project_t.txt
```

The output provides detailed information about each file in tabular format, so it is recommended to read it from top to bottom, left to right:

- 1st column: Displays file type (1st character, where `d` indicates directory and `-` indicates regular file) followed by file permissions (remaining 9 characters) for user, group, and other
- 2nd column: Shows the number of hard links pointing to the file
- 3rd column: Displays the owner name
- 4th column: Displays the group name
- 5th column: Indicates file size in bytes
- 6th-8th columns: Show the date and time the file was last modified
- Last column: Shows the filename

Based on the output of this command, there are one directory and four regular files. Hidden files (those starting with a dot) aren't displayed here because the command does not include the `-a` flag.

Describe the permissions string

Let's take an example of `project_k.txt` shown in line 2:

```
-rw-rw-rw- 1 researcher2 research_team   46 Jan 21 07:47
project_k.txt
```

As stated above, the first 10 characters represent the file type and user permissions with the first character representing file type where `d` tells us project_k.txt is a directory and **-** tells us this file is a regular file. The next 9 characters are divided equally into 3 categories: user, group, and other, respectively, with each category containing exactly 3 characters. Characters 2-4 represent user permissions, characters 5-7 represent group permissions, and characters 8-10 represent other permissions. The 1st character in any one category displays the read permission where r denotes read permission is granted and `-` denotes read permission is **not** granted. The 2nd character displays the write permission where `w` denotes write permission is granted and `-` denotes write permission is **not** granted. The 3rd character displays the execute permission where `x` denotes execute permission is granted and `-` denotes execute permission is **not** granted.

For example, in line 2, user (`researcher2`), group (`research_team`), and `other` (those who are not in either user or group categories) have read and write permissions for the `project_k.txt`. But none of them have execute permissions. This makes sense because the file is merely a text file denoted by `.txt` which should contain plain text and not executable code. However, allowing write permissions to `other` users creates a security risk, as anyone on the system can modify this file.

Change file permissions

The organization determined that the `other` category should never have write access to any of their files on the system. Referring to the organizational needs and the current file permissions, I determined that I need to remove write permissions from `other` users for `project_k.txt.`

The following code demonstrates how I used the `chmod` command to change `project_k.txt` permissions:

```
researcher2@08a5f572a345:~/projects$ chmod o-w project_k.txt
researcher2@08a5f572a345:~/projects$ ls -l project_k.txt
-rw-rw-r-- 1 researcher2 research_team 46 Jan 21 08:57 project_k.txt
```

As shown in the output above the `chmod` command successfully changed permissions for the `other` users. In the above case, this command is run to remove write permissions from the others. In the first argument, `o-w`, I specified the 1st character to indicate changes will be made to the `other` category. The 2nd character is a minus sign indicating I will remove permission(s) from `other` users. The 3rd character specifies write permissions, which is to be removed from the others. The second argument is the file name to change these permissions on. Removing write permissions for `other` users prevents unauthorized modifications to sensitive research files, reducing the risk of data tampering or corruption.

Change file permissions on a hidden file

The research team at the organization archived `project_x.txt`. Technically, they hid this file by prepending a period (`.`) to the file name: I.e. `.project_x.txt`. They determined that they do not want anyone to have write access to this archived project. However, the user and group should have read access.

The following code demonstrates how I used the `chmod` command to change permissions for `.project_x.txt`:

```
researcher2@08a5f572a345:~/projects$ ls -l .project_x.txt
-rw--w---- 1 researcher2 research_team 46 Jan 21 08:57 .project_x.txt
researcher2@08a5f572a345:~/projects$ chmod u=r,g=r .project_x.txt
researcher2@08a5f572a345:~/projects$ ls -l .project_x.txt
-r--r----- 1 researcher2 research_team 46 Jan 21 08:57 .project_x.txt
```

First, I displayed the file permissions for `.project_x.txt` using the `ls -l` command. Using the `chmod` command, I changed the user and group permissions. The first argument `u=r,g=r` specifies that u (representing user) and g (representing group) should have their permissions set to read-only. The last argument is the name of the file to change permissions on. I then displayed the file permissions again to confirm the changed permissions of `.project_x.txt`. Additionally, when verifying file permissions of other hidden files, we can use the `ls -la` command to display all files, including hidden files, in long format.

Change directory permissions

Furthermore, the organization only wants the user `researcher2` to have access to the drafts directory, including its contents. This simply means that the organization wants no one other than `researcher2` to have execute permissions. Execute permission for directories is crucial as it allows users to access the contents within the directory; without it, users cannot `cd` into the directory or access files inside it, even if they have permissions on those files.

The following code demonstrates how I used `chmod` to change the `drafts` directory permissions:

```
researcher2@1e98c1c7a813:~/projects$ chmod g-x drafts/
researcher2@1e98c1c7a813:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 22 01:39 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 22 02:06 ..
-rw--w---- 1 researcher2 research_team   46 Jan 22 01:39 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Jan 22 01:39 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan 22 01:39 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 22 01:39 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 22 01:39 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 22 01:39 project_t.txt
```

Since the user `researcher2` already has execute permissions of the `drafts` directory and `other` users have no execute permissions, I only need to remove the execute permission granted to users belonging to the `research_team` group. Since the group only has execute permissions I inputted the first argument as `g-x` which removes execute permissions from the group. These changes are then verified with the `ls -la` command.

As a side note, notice I added the `a` argument to the `ls` command. The output generates a list of all visible and hidden files, including the current directory and parent directory denoted by `.` and `..`, respectively.

Summary

To conclude, I looked into a hypothetical organization's file system and described how to read file type and permissions using the `ls -l` and `ls -la` commands. As a bonus, I also clarified what the remainder of a given line, outputted by these commands, represents. Lastly, I set the level of authorization of the system appropriately by modifying file permissions for user, group, and other categories for a single directory, regular files, and a hidden file using the `chmod` command. By implementing these permission changes, I applied the principle of least privilege to ensure users only have the minimum access levels required for their roles, which is a fundamental security practice that reduces the attack surface of the system.