File permissions in Linux

Scenario

Review the scenario below. Then, complete the step-by-step instructions.

You are a security professional at a large organization. You mainly work with their research team. Part of your job is to ensure users on this team are authorized with the appropriate permissions. This helps keep the system secure.

Your task is to examine existing permissions on the file system. You'll need to determine if the permissions match the authorization that should be given. If they do not match, you'll need to modify the permissions to authorize the appropriate users and remove any unauthorized access.

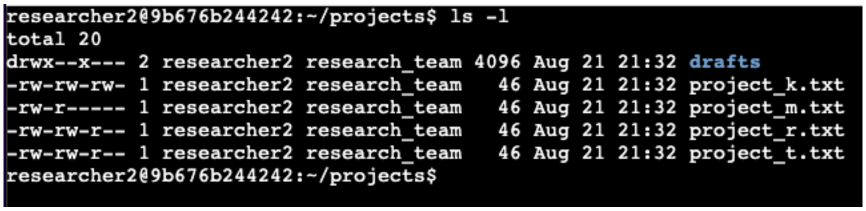
Note: This scenario involves investigating and updating the same file permissions as the ones in the Manage authorization lab. You can revisit the lab to get screenshots to include in your portfolio document. If you choose, it's also possible to complete this activity without revisiting the lab by typing your commands in the template.

Project description

As a security professional in a major organization's research team, I am responsible for maintaining a safe system by managing user authorizations and rights. My job entails meticulously analyzing the existing permissions on the file system to verify they correspond to the proper user access levels. This procedure necessitates a thorough assessment of each file's rights, including read, write, and execute capabilities for various user categories such as owners, groups, and others. In circumstances when permissions do not match the required authorization, I methodically change the permissions to establish accurate access controls.

Check file and directory details.

Command to check details: ls -l



```
researcher2@9b676b244242:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
researcher2@9b676b244242:~/projects$
```

The command "ls -l" is use to display a detailed list of files and directories in the current working directory. Unlike a simple directory listing provided by the "ls" command, the "-l" flag enhances the output by showing additional information. When executed, "ls -l" presents a structured output that includes file and directory names, along with details such as permissions, ownership, size, creation or modification timestamps, and group associations. This information is essential for users and administrators to gain a comprehensive understanding of the contents and attributes of files and directories within the specified directory.

Describe the permissions string.

Let’s take “drwxrwxrwx” as an example of permission string.

The string "drwxrwxrwx" represents the permissions and other information of a directory in a Unix-like operating system. Let's break down what each character and its meaning is:

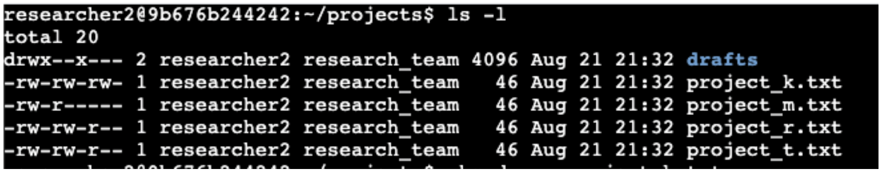
1. **d**: This letter signifies that it's a directory. If it were a regular file, this would be represented by a "-" (dash) instead.
2. **rwx**: The first set of three characters after the "d" represents the permissions for the owner of the directory. In this case, "rwx" means that the owner has read (r), write (w), and execute (x) permissions. Read permission allows the owner to view the contents of the directory, write permission allows them to modify the directory's contents, and execute permission allows them to enter the directory (if it's a script or program, for example).
3. **rwx**: The second set of three characters represents the permissions for the group associated with the directory. Similarly, "rwx" indicates that the group members also have read, write, and execute permissions for the directory.
4. **rwx**: The third set of three characters represents the permissions for others, i.e., anyone who is not the owner of the directory and not in the group associated with the directory. Once again, "rwx" means that these others have read, write, and execute permissions for the directory.

Form above example, the directory has the following permissions:

- Owner: Read, write, and execute permissions
- Group: Read, write, and execute permissions
- Others: Read, write, and execute permissions

Change file permissions.

Before permission change:



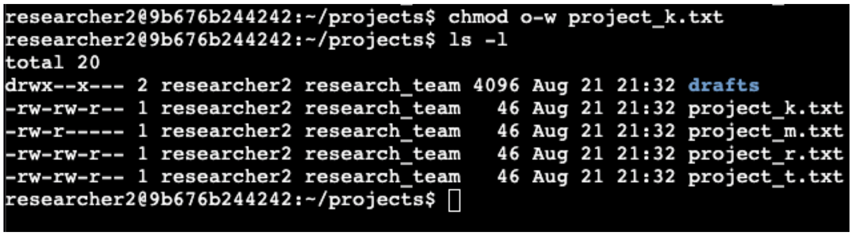
```
researcher2@9b676b244242:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
```

Let remove the write permission to the owner type of other from project_k.txt file.

Command: chmod o-w project_k.txt

Where – is use for removing permission and + is use for adding permission

After permission change:

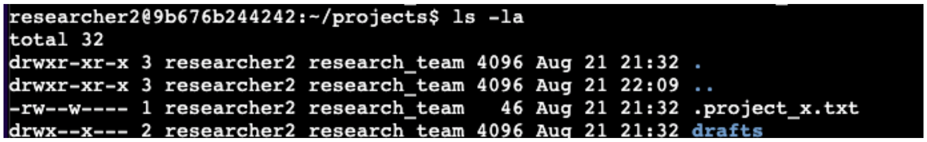


```
researcher2@9b676b244242:~/projects$ chmod o-w project_k.txt
researcher2@9b676b244242:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
researcher2@9b676b244242:~/projects$ ▯
```

In this command "chmod o-w project_k.txt", "chmod" is the directive used to modify permissions, while "o-w" signifies the action being taken. The "o" designates "others," referring to users who are neither the file's owner nor part of its associated group. The "-w" element represents the removal of the write permission. Consequently, executing this command would result in the removal of write permission for users categorized as "others" on the specified file "project_k.txt." This can help enhance security by restricting write access to those not directly associated with the file or its group.

Change file permissions on a hidden file.

Hidden file represented as .filename
Command to display hidden file: ls -la

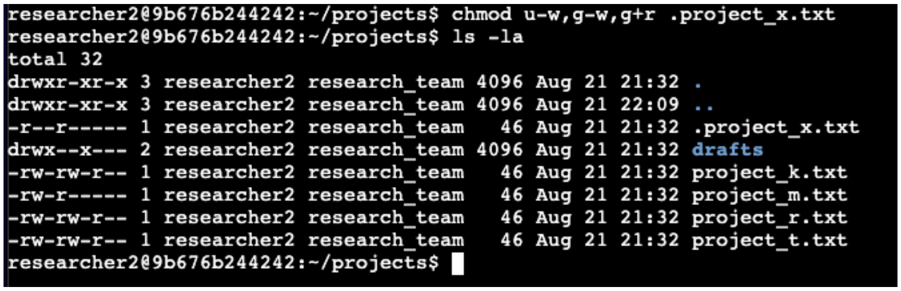


```
researcher2@9b676b244242:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 21 21:32 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 21 22:09 ..
-rw--w---- 1 researcher2 research_team   46 Aug 21 21:32 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
```

Let’s change the permissions of the hidden file .project_x.txt so that both the user and the group can read, but not write to, the file.

Command: chmod u-w,g-w,g+r .project_x.txt

Comma (,) is use to separate these permission changes, perform multiple modifications with a single command.



```
researcher2@9b676b244242:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@9b676b244242:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 21 21:32 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 21 22:09 ..
-r--r----- 1 researcher2 research_team   46 Aug 21 21:32 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
researcher2@9b676b244242:~/projects$ 
```

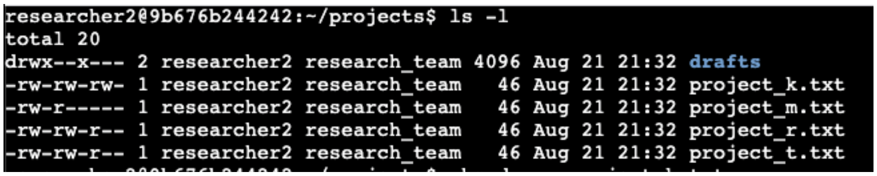
In this “chmod u-w,g-w,g+r .project_x.txt” command, the changes are made to three specific permission categories for the file. Firstly, the write permission for the owner (u) is revoked, ensuring that the owner cannot modify the file. Secondly, the write permission for the group (g) is also eliminated, prohibiting group members from making changes. Lastly, read permission for the group is added, enabling group members to access and read the contents of the file. This command streamlines the modification of multiple permissions in a single action, enhancing efficiency in controlling file access and security, while maintaining a concise syntax.

Change directory permissions

Let’s remove the execute permission for the group from the `drafts` directory.

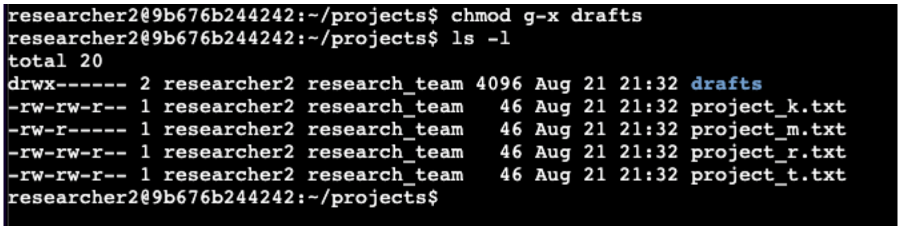
Command: chmod g-x drafts

Before permission change:



```
researcher2@9b676b244242:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
```

After permission change:



```
researcher2@9b676b244242:~/projects$ chmod g-x drafts
researcher2@9b676b244242:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Aug 21 21:32 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 21 21:32 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 21 21:32 project_t.txt
researcher2@9b676b244242:~/projects$
```

The command "chmod g-x drafts" is use to alter the permissions of a directory named "drafts." Specifically, the "chmod" command is used to modify permissions, and in this instance, it focuses on the "g-x" portion. This directive signifies the removal ("-") of the execute ("x") permission for the group ("g") associated with the specified file or directory. By executing this command, the group members' ability to execute (or run) the file as a script or program is restricted, while read and write permissions may remain unchanged. This command empowers users to finely control access rights to files, bolstering security and maintaining the desired level of user interaction.

Summary

I reviewed and modified file permissions to match authorized access. I checked that the proper users have suitable privileges by reviewing current permissions. When permissions did not match the required authorization, I used commands to alter access privileges, preventing any unwanted entrance. This proactive strategy assisted in safeguarding sensitive information for the firm while maintaining a secure and controlled file system environment.