REPORT
Linux File Permissions Management

v1.0.2

Author:

Eldon Gabriel

July 9, 2025

TABLE OF CONTENTS

REVISION HISTORY

Version	Date	Author	Description of Changes
v1.0.0	02/18/2025	Eldon G.	Initial draft.
v1.0.1	06/12/2025	Eldon G.	Added cover page, table of contents, & conclusion.
V1.0.2	07/09/2025	Eldon G.	Edited grammar and font size.

EXECUTIVE SUMMARY

As a cybersecurity professional working within a large organization, my primary responsibility is to ensure the security of the company's systems, specifically for the research team I support. My role involves maintaining access controls by reviewing and updating file permissions within the organization's file system. Secure file permissions are crucial to ensuring that sensitive research data is accessible only to authorized users and that unauthorized access is strictly prevented.

The research team needed to update the file permissions for specific files and directories within the `projects` directory, as the current permissions were not aligned with the required level of access. This posed a security risk, as it potentially allowed unauthorized users access to confidential research data. My task was to examine the existing permissions, determine where they did not match the necessary authorization, and modify them to ensure the correct access controls were in place.

To accomplish this, I performed the following tasks:

- **Evaluating Existing Permissions:** I examined the permissions on files and directories within the `projects` directory to ensure they aligned with the organization's security policies.
- **Modifying Permissions:** I used Linux commands such as `ls -la` to view existing permissions and `chmod` to update them, ensuring that the research team members had the appropriate level of access while preventing unauthorized users from accessing sensitive files.
- **Improving Security:** I specifically focused on sensitive files, hidden files, and restricted directories, ensuring only authorized team members had read, write, or execute permissions.

This process ensured the system remained secure, with proper authorization granted to users and unauthorized access prevented, strengthening the overall security posture of the organization.

SECTION 1.0: MANAGING LINUX PERMISSIONS

1.1 Reviewing File and Directory Attributes

The code below shows how I used Linux commands to check the existing permissions for a specific directory in the file system.

```
researcher2@de18f2d4a60a:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Feb  8 00:55 .
drwxr-xr-x 3 researcher2 research_team 4096 Feb  8 01:56 ..
-rw--w---- 1 researcher2 research_team   46 Feb  8 00:55 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Feb  8 00:55 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 00:55 project_k.txt
-rw------- 1 researcher2 research_team   46 Feb  8 00:55 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 00:55 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 00:55 project_t.txt
```

The first line of the screenshot shows the command I entered, while the following lines display the output. The code lists all the contents of the `projects` directory. I used the `ls -la` command to generate a detailed list of files, including hidden ones. The output indicates one directory named `drafts`, one hidden file named `.project_x.txt`, and five other project files. The first column has a 10-character string that represents the permissions set for every individual file or directory

1.2 Understanding the Permissions String

The 10-character string can be deconstructed to determine who is authorized to access the file and their specific permissions. The characters and what they represent are as follows:

- **1st character**: This character is either a `d` or a hyphen (`-`) and indicates the file type. If it's a `d`, it's a directory. If it's a hyphen (`-`), it's a regular file.
- **2nd-4th characters**: These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the user. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted to the user.
- **5th-7th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the group. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted for the group.
- **8th-10th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for others. This owner type consists of all other users on the system apart from the user and the group. When one of these characters is a hyphen (`-`) instead, that indicates that this permission is not granted for others.

For example, the file permissions for `project_t.txt` are `-rw-rw-r--`. The first character is a hyphen (`-`), which indicates that `project_t.txt` is a file. The second, fifth, and eighth characters are all `r`, meaning the user, group, and the others have read permissions. The third and sixth characters are write (`w`), indicating that only the user and group have write permissions. While no one has executed permissions for `project_t.txt`.

1.3 Modifying File Permissions

The company determined that the others shouldn't have write access to any files. To comply, I referred to the file permissions that I previously returned. I figured `project_k.txt` must have write access removed for others.

The following code shows how I used Linux commands to accomplish this:

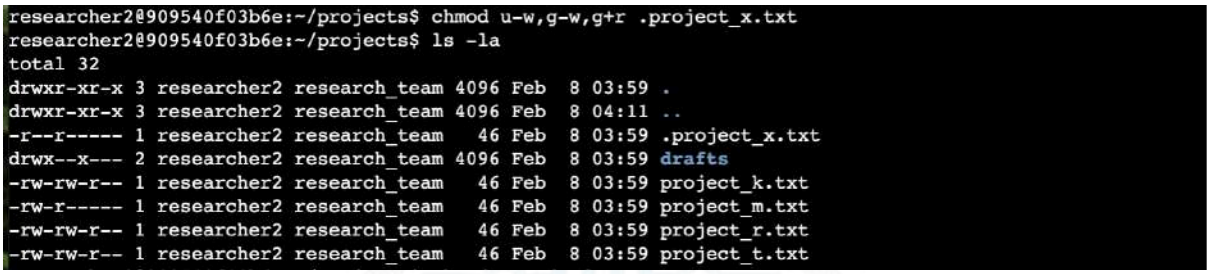
```
researcher2@5176a51ef052:~/projects$ chmod o-w project_k.txt
researcher2@5176a51ef052:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Feb  8 03:14 .
drwxr-xr-x 3 researcher2 research_team 4096 Feb  8 03:25 ..
-rw--w---- 1 researcher2 research_team   46 Feb  8 03:14 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Feb  8 03:14 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 03:14 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Feb  8 03:14 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 03:14 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Feb  8 03:14 project_t.txt
researcher2@5176a51ef052:~/projects$
```

See, the first two lines of the screenshot? This shows the commands I entered, while the remaining lines display the results of the second command. The `chmod` command is used to change the permissions of files and directories. The first argument specifies which permissions to modify, and the second argument identifies the file or directory. In this case, I canceled the write permissions for the 'others' on the `project_k.txt` file. After that, I ran the `ls -la` command to verify the changes I made.

1.3.1 Adjusting Permissions on Hidden Files

The research team recently archived `project_k.txt` and decided that no one should have write access to the file. However, both the user and group should retain read access.

The following code shows how I used Linux commands to accomplish this:

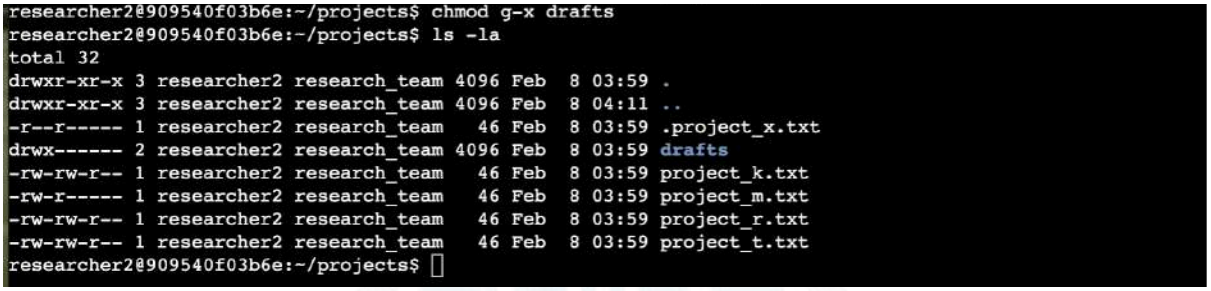


The first two lines of the screenshot show the commands I entered, while the remaining lines display the output from the second command. I identified `project_k.txt` as a hidden file because it begins with a period (`.`). In this example, I removed write permissions from both the user and the group. Specifically, I used `u-w` to remove write permissions from the user and `g-w` to remove write permissions from the group. Additionally, I added read permissions to the group using `g+r`.

1.4 Configuring Directory Permissions

The company only wants the `researcher2` user to have access to the `drafts` directory. This means to ensure that only the user `researcher2` should have access to execute permissions.

The following code shows how I used Linux commands to accomplish this:



The output shows the permission listing for several files and directories. Line 1 represents the current directory, "projects," while line 2 shows the parent directory, "home." Line 3 displays a regular file named `.project_x.txt`, and line 4 shows the directory "drafts" with restricted permissions. In this case, only the user `researcher2` has execute permissions. Previously, the group also had execute permissions, so by using the `chmod` command, it removed them. Since `researcher2` already had execute permissions, no changes were necessary for that user.

SECTION 2.0: CONCLUSION

2.1 Summary

As a cybersecurity professional supporting the research team, I was responsible for securing Linux file system permissions to prevent unauthorized access and ensure compliance with organizational policies. My task involved reviewing permission settings in the projects directory, identifying discrepancies, and applying appropriate access controls.

2.2 Real-World Actions and Impact

Using Linux commands such as `ls -la` and `chmod`, I evaluated and updated permissions on sensitive files and directories. For instance:

- I removed write access from unauthorized users on critical files.
- I restricted access to hidden files like `.project_x.txt`, ensuring only read access for the appropriate users.
- I secured the `drafts` directory, limiting access exclusively to the `researcher2` user.

These actions aligned with the principle of least privilege and protected sensitive research assets from unintended exposure or modification.

2.3 Recommendations and Best Practices

To maintain a strong security posture, I recommend the following:

- Perform regular permission audits using tools like `find`, `stat`, and `getfacl`.
- Avoid overly permissive settings like `777` on shared directories or sensitive files.
- Educate team members on proper file storage and permission hygiene.
- Implement automated alerts for permission changes in high-risk directories.

By enforcing proper file and directory permissions, organizations can significantly reduce the risk of data breaches and maintain operational integrity.