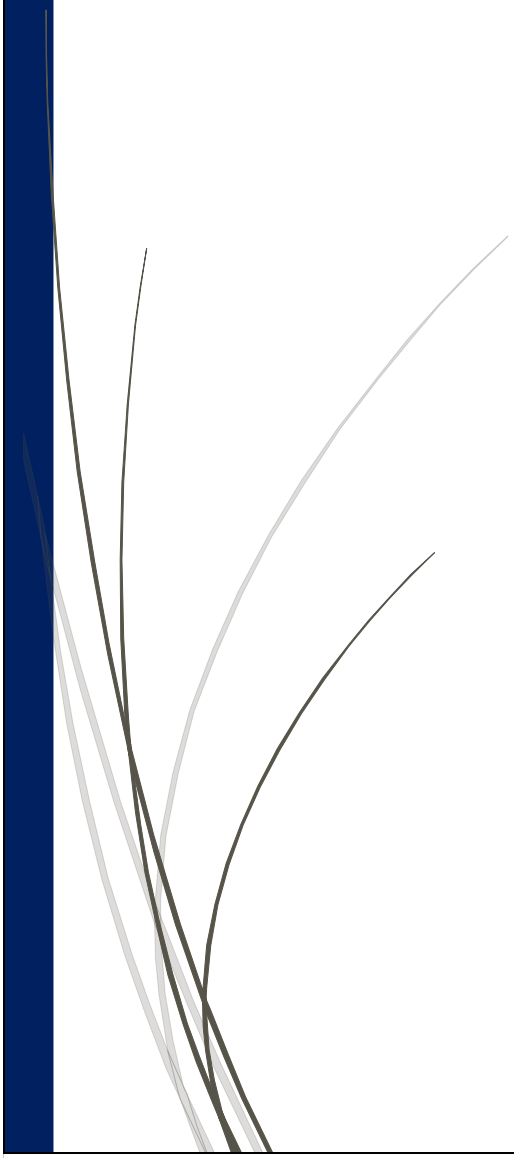
5/1/2025

# LINUX PERMISSIONS AND DATA SECURITY

RESEARCH REPORT



Balaji Varaprasad
CYBER SAPHIENS

## INTRODUCTION:

Linux is a powerful and secure operating system widely used in servers, enterprises, and cybersecurity. One of its core strengths is the user management and permissions system, which ensures that only authorized user can access and modify files and processes. These permissions help ensures that users and processes only access data they are authorized to, protecting the system from accidental or malicious changes.

## USER MANAGEMENT IN LINUX:

User management in Linux involves creating, modifying, and deleting user accounts, as well as assigning them appropriate permissions and group memberships. It's essential for security, multi-user access, and system organization.

## USER TYPES:

| TYPE | DESCRIPTION |
|---|---|
| Root | The superuser with full system control. Can modify any file and can execute any commands (UID 0). |
| System users | Users created by the system for running services like databases and web servers. |
| Regular users | Created by the root user, these users have restricted permissions and limited system access. |

## USER MANAGEMENT COMMANDS:

Here are some common user management commands:

- To add a user (creates a new user with home directory and default settings)

```
sudo adduser username
```

- Delete a User

```
sudo deluser username
```

- Modify a user and add a user to a group

```
sudo usermod –aG groupname username
```

- Change Password

```
sudo passwd username
```

- Add a group

```
sudo addgroup groupname
```

- Check Group Membership

```
groups username
```

- To list all users currently logged into the system

```
who
```

- To list all users on the system

```
getent passwd
```

While these are some of the most commonly used commands, Linux offers a wide range of additional commands and configuration options for more advanced user and permission management tasks. Exploring these further can help ensure better control and security in multi-user environments.

## UNDERSTANDING LINUX PERMISSIONS:

Linux uses a permission-based system that assigns permissions to three types of users:

- **Owner** – The person who created the file or was assigned ownership(u).
- **Group** – A set of users who are part of the file's assigned group(g).
- **Others** – All other users on the system(o).

## TYPES OF PERMISSIONS:

| SYMBOL | PERMISSIONS | APPLIES TO | DESCRIPTION |
|---|---|---|---|
| **r** | Read | Files and Directories | For files: view contents.<br>For directories: list files. |
| **w** | Write | Files and Directories | For files: modify contents.<br>For directories: add/remove/rename files. |
| **x** | Execute | Files and Directories | For files: run the file as a program/script.<br>For directories: enter (cd) |

## TO VIEW PERMISSIONS:

Use the ls -l command to view permissions:

```
ls -l file.txt
```

Expected output:

```
-rw-r--r-- 1 user group 1024 May 1 16:41 file.txt
```

```
-rw-r--r-- 1 Panda cysec 1024 May 1 16:41 file.txt
```

Breakdown of output:

**-rw-r--r--** ➡ Permissions

**-** ➡ Regular file (or d for directory)

**rw-** ➡ Owner (read/write)

**r--** ➡ Group (read-only)

**r--** ➡ Others (read-only)

**Balaji**: Owner

**cysec**: Group

## IMPORTANCE OF FILE OWNERSHIP AND chmod COMMAND:

File ownership in Linux is essential for keeping the system secure, organized, and fair for everyone using it. When a file is created, it's automatically assigned to the user who made it (the owner) and also linked to a group. This ownership helps the system decide who can read, write, or execute the file. It ensures that personal files stay private and that system files aren't accidentally changed by someone who shouldn't have access. In shared environments, like offices or servers, group ownership also makes collaboration easier by letting team members work on common files without exposing them to everyone else.

The chmod command plays a big role in managing access to these files by letting you change their permissions. Think of it like setting rules for your stuff who can open it, who can change it, and who can use it. With chmod, users and administrators can give or restrict access as needed, whether it's allowing a teammate to edit a file or making a script executable. This flexibility is key to maintaining both security and productivity, allowing users to safely share what they need while keeping everything else protected.

## FILE OWNERSHIP:

## TO CHANGE FILE OWNERSHIP:

Every file or directory in Linux has two key types of ownership: the owner and the group. The owner is typically the user who created the file and has primary control over it, including the ability to read, write, or execute the file depending on the set permissions. The group refers to a collection of users who are granted shared access to the file. This setup allows Linux to manage permissions efficiently by giving one level of control to the individual who owns the file and another level to a group of users who may need shared access for collaboration, all while maintaining system security and user privacy.

Change ownership using:

```
sudo chown Panda:cysec myfile.txt
```

(Now Panda is the owner and cysec is the group.)

## THE chmod COMMAND (CHANGE MODE):

The chmod command is used to modify permissions. Basically, there are two types of modes:

- Symbolic mode
- Numeric mode

**Symbolic Mode:**

```
chmod u+x myscript.sh
```

(Add execute for owner)

```
chmod g-w myfile.txt
```

(Remove write from group)

```
chmod o=r myfile.txt
```

(Set read-only for others)

**Numeric Mode:**

In numerical mode each permissions has a number:

| Permission | Value |
|---|---|
| Read (r) | 4 |
| Write (w) | 2 |
| Execute (x) | 1 |

When combining them:

7 (r+w+x)

6 (r+w)

5 (r+x)

4 (r)

Command format:

```
chmod 755 myscript.sh
```

Means:

**7 ➡** Owner- Read/Write/Execute

**5 ➡** Group- Read/Execute

**5 ➡** Others- Read/Execute

**Tabular representation of symbolic and numerical commands:**

Common numerical permissions:

| Permission | Owner | Group | Others | Use Case | chmod Command |
|---|---|---|---|---|---|
| 755 | rwx | r-x | r-x | Public scripts, readable by all | `chmod 755 file.sh` |
| 700 | rwx | --- | --- | Private scripts or folders | `chmod 700 secrets/` |
| 644 | rw- | r-- | r-- | Public text files (no edits) | `chmod 644 file.txt` |
| 600 | rw- | --- | --- | Private files (e.g., SSH keys) | `chmod 600 id_rsa` |
| 775 | rwx | rwx | r-x | Shared group access | `chmod 775 shared/` |
| 770 | rwx | rwx | --- | Team-only folder (no outside access) | `chmod 770 team/` |
| 444 | r-- | r-- | r-- | Read-only for everyone | `chmod 444 policy.md` |

Symbolic Mode:

| Symbol | Meaning | Example | Description |
|---|---|---|---|
| u | user (owner) | `chmod u+x file` | Add execute for owner |
| g | group | `chmod g-w file` | Remove write for group |
| o | others | `chmod o=r file` | Set read-only for others |
| a | all (user+group+others) | `chmod a+x script.sh` | Make script executable for everyone |
| + | Add permission | `chmod g+w shared.txt` | Allow group to write |
| - | Remove permission | `chmod o-x test.sh` | Remove execute for others |
| = | Set exact permission | `chmod u=rw file.txt` | User gets read+write, nothing else |

## IMPORTANCE OF SETTING PROPER AND APPROPRIATE PERMISSIONS:

Setting the right permissions is super important and crucial for security and data integrity. Because it helps protect your system from accidental damage or unauthorized access. Here are a few examples where setting appropriate permissions is absolutely crucial:

**1) Protecting system configuration files:**

Example: `/etc/passwd, /etc/shadow`

These files store sensitive information like user account details and encrypted passwords.

- `/etc/passwd` should be readable but not writable by regular users.
- `/etc/shadow` should only be accessible by the root user.

```
ls -l /etc/shadow
-r-------- 1 root root 1000 May 1 18:23 /etc/shadow
```

**2) Restricting Access to User Home Directories:**

Example: **/home/username**

```
chmod 700 /home/panda
```

Users shouldn’t be able to read or modify each other's files unless specifically allowed. This ensures only panda can access his personal files.

**3) Securing Web Server Files:**

Example: `/var/www/html`

In a web hosting setup, files served by the web server must be readable by the server but not writable by others to avoid code injection attacks.

```
chmod 644 index.html
```

This means owner can read/write, others can only read.

**4) Shared Project Folders:**

Example: `/project/team`

```
chown -R :cysec /project/team
chmod -R 770 /project/team
```

In a collaborative environment, you might want everyone in a group to access and edit files, but prevent outside users from accessing them. Only team members in the **cysec** group get full access.

**5) Making Scripts Executable:**

Example: `backup.sh`

```
chmod 750 backup.sh
```

A script should be executable but not writable by everyone, or someone could alter it to run malicious commands. Only the owner and group can run it, the rest can't even touch it.

## CONCLUSION:

Setting the right permissions in Linux is like putting locks on important doors, you decide who gets in, who can make changes, and who can only look. For instance, system files like password databases contain sensitive information and must be protected. If a regular user accidentally (or intentionally) gained write access to these files, it could compromise the entire system. Similarly, in personal or professional environments, private documents, SSH keys, and financial records need to be set with restricted permissions so only the rightful owner can access or modify them. Without these restrictions, critical data could be lost, corrupted, or stolen.

Permissions are also vital when sharing files in a group setting. Imagine a team working on a software project where each member can access shared code, but no one outside the team should touch it. Proper group permissions ensure smooth collaboration while keeping the project safe from unintended edits or deletions. By carefully setting permissions, you're not just protecting data you're creating a safe, efficient space for users, teams, and systems to operate without stepping on each other's toes.

**REFRENCES:**

https://www.webasha.com/blog/managing-users-and-permissions-in-linux-a-complete-guide-to-secure-access-control#sec4

https://www.geeksforgeeks.org/users-in-linux-system-administration/

# THANK YOU