User and Security Management in Linux

Scenario Overview

- Create a shared project directory and apply SGID so that all new files inside inherit the group ownership (webdev).
- Set a sticky bit on a temporary directory to restrict file deletion.
- Force a password reset for the intern01 user.
- Apply an extended attribute (immutable flag) on an important configuration file.
- Use an Access Control List (ACL) to grant extra permissions to a user on a file.
- Demonstrate a simple polkit (pkexec) example to run a command with elevated privileges without sudo.
- At each step, you will verify the changes by checking relevant files/directories or running commands to confirm the new settings.

1. Create a Shared Directory for webdev with SGID

Goal: Any file created within this directory automatically belongs to the webdev group.

Check existing groups:
tiago-paquete@tiago-paquete-Linux:~$ **cut -d: -f1 /etc/passwd | tail -n 5**
==========
tiago-paquete
sshd
alice
bob
intern01
==========

cut -d: -f1
cut: A command to extract specific sections of text.
-d:: Sets the delimiter to : (colon).
-f1: Tells cut to extract the first field, which is the username.
So this part extracts just the usernames from the file.

|
A pipe: Passes the output of the first command (cut) into the second command.

tail -n 3
tail: Outputs the last part of a file or input.
-n 3: Means "show the last 3 lines".

So this part shows the last 3 usernames from the list.

Create the directory:
sudo mkdir /srv/webdev_shared

tiago-paquete@tiago-paquete-Linux:~$ **sudo mkdir /srv/webdev_shared**
==========
[sudo] password for tiago-paquete:
==========

/srv/webdev_shared:
- **/srv** is a standard directory used to hold data for services (like web servers, FTP, etc.).
- **webdev_shared** is the name of the directory being created — intended for shared web development files.

Check:
tiago-paquete@tiago-paquete-Linux:~$ **ls -ld /srv/webdev_shared**
==========
drwxr-xr-x 2 root root 4096 May 2 09:26 **/srv/webdev_shared**
==========

-d: Lists the directory itself rather than its contents.

Set the group to webdev:
sudo chown :webdev /srv/webdev_shared

Apply SGID so that newly created files inherit the webdev group:
sudo chmod 2775 /srv/webdev_shared

The leading 2 in 2775 sets the SGID bit.

Numeric Permission Codes for `chmod`:

Digit	Binary	Special Bit	Description
0	0	None	No special permissions
1	1	Sticky Bit	For directories: only the file owner or root can delete files inside
2	10	SGID	Files: run with group's privileges; Directories: files inherit group
4	100	SUID	Files: run with owner's privileges (useful for programs like `passwd`)

Check ownership and permissions:
ls -ld /srv/webdev_shared

tiago-paquete@tiago-paquete-Linux:~$ sudo chown :webdev /srv/webdev_shared
tiago-paquete@tiago-paquete-Linux:~$ sudo chmod 2775 /srv/webdev_shared
tiago-paquete@tiago-paquete-Linux:~$ ls -ld /srv/webdev_shared
==========
drwxrw**s**r-x 2 root webdev 4096 May 2 09:26 /srv/webdev_shared
==========

Field	Meaning
d	It's a directory.
rwxrwsr-x	Permissions:
	- `rwx` → owner (`root`) can read/write/execute
	- `rws` → group (`webdev`) has rwx, **s** means SGID is set
	- `r-x` → others can read and execute
2	Number of hard links (usually 2 for an empty directory)
root	Owner of the directory
webdev	Group of the directory
4096	Size of the directory (in bytes)
May 2 09:26	Last modified date/time
/srv/webdev_shared	Directory path

As alice or bob, create a file inside /srv/webdev_shared:

```
su - alice
cd /srv/webdev_shared
touch testfile
ls -l testfile
```

```
tiago-paquete@tiago-paquete-Linux:~$ su - Alice
==========================================================================
Password:
==========================================================================
alice@tiago-paquete-Linux:~$ cd /srv/webdev_shared
alice@tiago-paquete-Linux:/srv/webdev_shared$ touch testfile
alice@tiago-paquete-Linux:/srv/webdev_shared$ ls -l testfile
==========================================================================
-rw-rw-r-- 1 alice webdev 0 May  2 09:53 testfile
==========================================================================
```

2. Set a Sticky Bit on a Temporary Directory

Goal: Restrict file deletion in a temporary folder so only file owners (or root) can remove their own files.

Create a directory
sudo mkdir /srv/tmp_staging

Set the permissions so it's open to everyone but with a sticky bit:
sudo chmod 1777 /srv/tmp_staging

1 in 1777 is the sticky bit.
Permissions become drwxrwxrwt.

Verification
Check permissions:
ls -ld /srv/tmp_staging

```
tiago-paquete@tiago-paquete-Linux:~$ sudo mkdir /srv/tmp_staging
tiago-paquete@tiago-paquete-Linux:~$ sudo chmod 1777 /srv/tmp_staging
tiago-paquete@tiago-paquete-Linux:~$ ls -ld /srv/tmp_staging
==========
drwxrwxrwt 2 root root 4096 May  2 09:59 /srv/tmp_staging
==========
```

The **t** at the end indicates the sticky bit.

Test by creating a file as alice, then trying to remove it as bob. Only alice (the owner) or root can delete it.

```
tiago-paquete@tiago-paquete-Linux:~$ su - Alice
==========
Password:
==========
alice@tiago-paquete-Linux:~$ cd /srv/tmp_staging
alice@tiago-paquete-Linux:/srv/tmp_staging$ touch testfile
alice@tiago-paquete-Linux:/srv/tmp_staging$ ls -l
==========
total 0
-rw-rw-r-- 1 alice alice 0 May  2 10:09 testfile
==========
alice@tiago-paquete-Linux:/srv/tmp_staging$ su - bob
==========
Password:
==========
bob@tiago-paquete-Linux:~$ cd /srv/tmp_staging
bob@tiago-paquete-Linux:/srv/tmp_staging$ ls -l
==========
total 0
-rw-rw-r-- 1 alice alice 0 May  2 10:09 testfile
==========
bob@tiago-paquete-Linux:/srv/tmp_staging$ rm testfile
==========
rm: remove write-protected regular empty file 'testfile'? y
rm: cannot remove 'testfile': Operation not permitted
==========
```

3. Force a Password Reset for intern01

Goal: Require intern01 to create a new password at next login (e.g., policy updates).

Expire the password immediately:

sudo passwd --expire intern01
or
sudo chage -d 0 intern01

1. sudo passwd --expire intern01

Command Purpose: Immediately expires the current password for intern01.
Effect: The next time intern01 logs in, they are required to set a new password.
How it works: It sets the account's password expiry date to the past (i.e., "expired now").
Use case: Straightforward, commonly used to enforce password resets for security policies.

2. sudo chage -d 0 intern01
Command Purpose: Sets the last password change date to 0 (the epoch).
Effect: Linux treats this as if the password has never been set, so a change is required immediately at next login.
Equivalent Outcome: Forces a password reset on next login — just like passwd --expire.

chage stands for change age (of password), and -d modifies the "last changed" date.

```
tiago-paquete@tiago-paquete-Linux:~$ sudo passwd --expire intern01
==========================================================================
passwd: password changed.
==========================================================================
```

Verification
Check the password status with chage:

sudo chage -l intern01

```
tiago-paquete@tiago-paquete-Linux:~$ sudo chage -l intern01
==========================================================================
Last password change                                    : password must be changed
Password expires                                        : password must be changed
Password inactive                                       : password must be changed
Account expires                                         : Dec 31, 2025
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
==========================================================================
```

On the next login, intern01 should be forced to set a new password.

4. Apply an Extended Attribute (Immutable Flag)

Goal: Protect an important config file from being altered or deleted, even by root (until the flag is removed).

Create or identify a config file (example /etc/critical.conf):
sudo touch /etc/critical.conf

tiago-paquete@tiago-paquete-Linux:~$ **sudo touch /etc/criticaltest.conf**
tiago-paquete@tiago-paquete-Linux:~$ **ls -l /etc/criticaltest.conf**
==========
-rw-r--r-- 1 root root 0 May 2 10:24 /etc/criticaltest.conf
==========

Set the immutable flag:
sudo chattr +i /etc/criticaltest.conf

chattr: Change file attributes on a **Linux file system** (typically ext2/3/4)
+i: Add the immutable attribute to the file.

What Does the +i (Immutable) Flag Do?
- When a file is set to immutable:
  - It cannot be modified, even by root.
  - It cannot be renamed or deleted.
  - No one can write to it, append to it, or truncate it.
  - Symbolic links pointing to it also cannot override this restriction.

It is a very strong protection for critical configuration files.

Before:
tiago-paquete@tiago-paquete-Linux:~$ **ls -l /etc/criticaltest.conf**
==========
-rw-r--r-- 1 root root 0 May 2 10:24 /etc/criticaltest.conf
==========
tiago-paquete@tiago-paquete-Linux:~$ **lsattr /etc/criticaltest.conf**
==========
----------e------- /etc/criticaltest.conf
==========

e: File is using extents for storage (normal for ext4)

After:
tiago-paquete@tiago-paquete-Linux:~$ **sudo chattr +i /etc/criticaltest.conf**
tiago-paquete@tiago-paquete-Linux:~$ **lsattr /etc/criticaltest.conf**
==========
----**i**---------e------- /etc/criticaltest.conf
==========

Try modifying or removing /etc/critical.conf:
sudo rm /etc/criticaltest.conf

tiago-paquete@tiago-paquete-Linux:~$ **rm /etc/criticaltest.conf**
==========
rm: cannot remove '/etc/criticaltest.conf': Operation not permitted
==========

Remove the flag if needed:
sudo chattr -i /etc/criticaltest.conf

tiago-paquete@tiago-paquete-Linux:~$ **sudo chattr -i /etc/criticaltest.conf**
tiago-paquete@tiago-paquete-Linux:~$ **lsattr /etc/criticaltest.conf**
==========
----------e------- /etc/criticaltest.conf
==========

5. Use ACLs to Grant Extra Permissions

Suppose you have a file in bob's home directory, /home/bob/secret_project.txt, and you need to give Alice read/write access, but don't want to change the file's group or standard permissions.

Create the file:
su - bob
touch ~/secret_project.txt

(Ensure the file is there. Ownership is bob:bob by default.)

tiago-paquete@tiago-paquete-Linux:~$ **su - bob**
==========
Password:
==========
bob@tiago-paquete-Linux:~$ **pwd**
==========
/home/bob
==========
bob@tiago-paquete-Linux:~$ **touch ~/secret_project.txt**
bob@tiago-paquete-Linux:~$ **ls -l**
==========
total 0
-rw-rw-r-- 1 bob bob 0 May 2 10:55 secret_project.txt
==========

Grant read and write to alice using ACL:
sudo setfacl -m u:alice:rw /home/bob/secret_project.txt

tiago-paquete@tiago-paquete-Linux:~$ **sudo setfacl -m u:alice:rw /home/bob/secret_project.txt**
==========
[sudo] password for tiago-paquete:
==========

Part	Meaning
sudo	Run with superuser privileges (required to modify files owned by others)
setfacl	Command to set file ACLs
-m	Modify the ACL
u:alice:rw	Give user `alice` read and write permissions
/home/bob/ secret_project.txt	The file on which ACL is applied

Verify:
getfacl /home/bob/secret_project.txt

tiago-paquete@tiago-paquete-Linux:~$ **getfacl /home/bob/secret_project.txt**
==========
getfacl: /home/bob/secret_project.txt: Permission denied
==========
tiago-paquete@tiago-paquete-Linux:~$ **sudo !!**
==========
sudo getfacl /home/bob/secret_project.txt
==========
getfacl: Removing leading '/' from absolute path names
file: home/bob/secret_project.txt
owner: bob
group: bob
user::rw-
user:alice:rw-
group::rw-
mask::rw-
other::r--
==========

Line	Meaning
Removing leading '/'	Cosmetic message: shown because `getfacl` is cleaning the path display. Doesn't affect functionality.
# file: home/bob/...	Relative path shown due to above behavior
# owner: bob	Owner of the file
# group: bob	Group associated with the file
user::rw-	Owner (`bob`) has read + write
user:alice:rw-	User `alice` is granted read + write via ACL
group::rw-	Group `bob` has read + write
mask::rw-	Maximum effective permissions for users/groups with ACL entries
other::r--	Everyone else can only read

6. Demonstrate polkit with pkexec

Goal: Run a command with elevated privileges without using sudo. pkexec is part of the PolicyKit (polkit) framework.

What is polkit?

Polkit (PolicyKit) is a toolkit used for fine-grained control of system-wide privileges.
It allows unprivileged users to run privileged actions without needing full sudo access.
It is especially useful in desktop environments (e.g., GNOME, KDE), but also works in server contexts.

Common Tools Involved

Tool	Purpose
pkexec	Run a command as root (like sudo, but uses polkit)
pkaction	List or inspect polkit actions
polkitd	The PolicyKit authentication agent (daemon)

Run a root-level command with polkit:
pkexec whoami

You will be prompted for an authorized user's password. If your user is configured in polkit to allow administrative tasks, the command will run as root.

```
tiago-paquete@tiago-paquete-Linux:~$ pkexec whoami
==========================================================================
Root
==========================================================================
```

This indicates the command ran with elevated privileges.
Note: If you are prompted with Authentication is needed to run /usr/bin/whoami' as the super user`, you must enter your (allowed) user's password.

Commands

Command	Description
cut -d: -f1 /etc/passwd	Extracts all usernames from `/etc/passwd`.
tail -n 5	Displays the last 5 lines of input (used here to show last 5 users).
sudo mkdir /srv/webdev_shared	Creates a new shared directory for the `webdev` group.
sudo chown :webdev /srv/webdev_shared	Changes the group ownership of the directory to `webdev`.
sudo chmod 2775 /srv/webdev_shared	Sets directory permissions with SGID so new files inherit the `webdev` group.
ls -ld /srv/webdev_shared	Displays permissions and ownership of the shared directory.
su - alice	Switches to user `alice` (useful for testing SGID and sticky bit behavior).
cd /srv/webdev_shared	Navigates to the shared directory.
touch testfile	Creates a test file to check group ownership inheritance.
ls -l testfile	Displays detailed info about the test file including group ownership.
sudo mkdir /srv/tmp_staging	Creates a temporary directory for sticky bit testing.
sudo chmod 1777 /srv/tmp_staging	Sets full access with sticky bit so only file owners can delete their files.
ls -ld /srv/tmp_staging	Verifies sticky bit is set (look for `t` at the end of permissions).
sudo passwd --expire intern01	Forces `intern01` to change password at next login.
sudo chage -d 0 intern01	Alternative way to expire password using `chage`.
sudo chage -l intern01	Lists password aging and expiration information for `intern01`.
sudo touch /etc/critical.conf	Creates a protected configuration file.
sudo chattr +i /etc/critical.conf	Applies the **immutable** attribute to prevent deletion/modification.
lsattr /etc/critical.conf	Verifies extended attributes like the immutable flag.
sudo chattr -i /etc/critical.conf	Removes the immutable attribute if needed.
su - bob	Switches to user `bob` to simulate collaboration with file access.
touch ~/secret_project.txt	Creates a file owned by `bob`.
sudo setfacl -m u:alice:rw /home/bob/secret_project.txt	Grants `alice` read/write access via ACL.
getfacl /home/bob/secret_project.txt	Views ACL entries on the file.
sudo setfacl -x u:alice /home/bob/secret_project.txt	Removes the ACL entry for `alice`.

pkexec whoami	Executes the `whoami` command as root using PolicyKit (GUI login session required).
which whoami	Shows the full path to the `whoami` command.