PERMESSI LINUX

Gestione dei permessi di lettura, scrittura ed esecuzione in Linux (chmod)

SCENARIO REALISTICO

Su un server Linux e si prepara una cartella “condivisa” per un team:

- **il proprietario (KALI)** possa leggere/scrivere/modificare tutto
- **il gruppo devteam** possa **leggere ed entrare** nella directory, ma **non scrivere**
- **gli altri utenti** non debbano vedere nulla

In più hai un file di script di deploy che:

- KALI puoi eseguire
- il team può leggere (per revisione)
- nessun altro deve accedere

Dettali:

Ho simulato una cartella di progetto “shared_reports” che contiene un report (report.txt) e uno script (deploy.sh). L’obiettivo è consentire al proprietario pieno controllo, al gruppo di lavoro sola consultazione, e impedire l’accesso ad utenti esterni.

Motivazione delle scelte dei permessi

- **Directory shared_reports impostata a 750 (rwxr-x---)**
    - Proprietario: rwx per gestire contenuti e struttura.
    - Gruppo: r-x per poter entrare nella directory e consultare i file, ma senza creare/modificare/cancellare contenuti (manca w).
    - Altri: --- per impedire visibilità e accesso ai contenuti.
- **File report.txt impostato a 640 (rw-r-----)**
    - Proprietario: lettura e scrittura per aggiornare il report.
    - Gruppo: sola lettura per consultazione.
    - Altri: nessun accesso.
- **File deploy.sh impostato a 740 (rwxr-----)**
    - Proprietario: può eseguire lo script (permesso x).
    - Gruppo: può leggerlo per revisione ma non eseguirlo (manca x) per evitare esecuzioni non autorizzate.
    - Altri: nessun accesso.

CONFIGURAZIONE PERMESSI

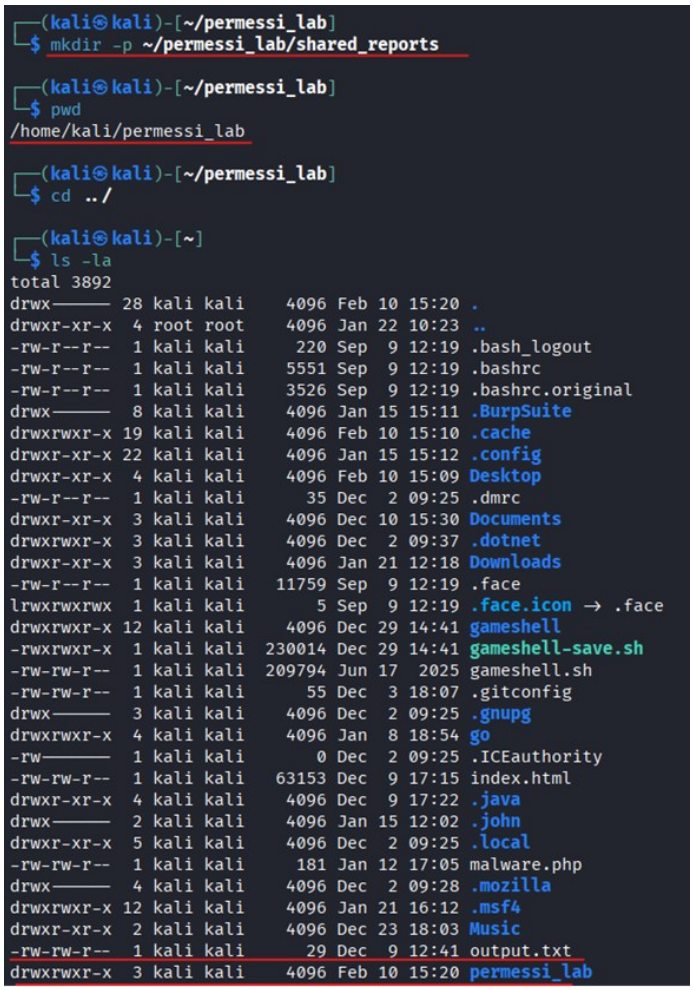
1) Screenshot: creazione file/directory

```
mkdir -p ~/permessi_lab/shared_reports
cd ~/permessi_lab

# file di report
echo "Report interno - Q1" > shared_reports/report.txt

# script “di deploy” finto
cat > shared_reports/deploy.sh <<'EOF'

#!/bin/bash
echo "Deploy simulato: OK"
EOF
```



```
┌──(kali㉿kali)-[~/permessi_lab]
└─$ mkdir -p ~/permessi_lab/shared_reports

┌──(kali㉿kali)-[~/permessi_lab]
└─$ pwd
/home/kali/permessi_lab

┌──(kali㉿kali)-[~/permessi_lab]
└─$ cd ../

┌──(kali㉿kali)-[~]
└─$ ls -la
total 3892
drwx------ 28 kali kali    4096 Feb 10 15:20 .
drwxr-xr-x  4 root root    4096 Jan 22 10:23 ..
-rw-r--r--  1 kali kali     220 Sep  9 12:19 .bash_logout
-rw-r--r--  1 kali kali    5551 Sep  9 12:19 .bashrc
-rw-r--r--  1 kali kali    3526 Sep  9 12:19 .bashrc.original
drwx------  8 kali kali    4096 Jan 15 15:11 .BurpSuite
drwxrwxr-x 19 kali kali    4096 Feb 10 15:10 .cache
drwxr-xr-x 22 kali kali    4096 Jan 15 15:12 .config
drwxr-xr-x  4 kali kali    4096 Feb 10 15:09 Desktop
-rw-r--r--  1 kali kali      35 Dec  2 09:25 .dmrc
drwxr-xr-x  3 kali kali    4096 Dec 10 15:30 Documents
drwxrwxr-x  3 kali kali    4096 Dec  2 09:37 .dotnet
drwxr-xr-x  3 kali kali    4096 Jan 21 12:18 Downloads
-rw-r--r--  1 kali kali   11759 Sep  9 12:19 .face
lrwxrwxrwx  1 kali kali       5 Sep  9 12:19 .face.icon → .face
drwxrwxr-x 12 kali kali    4096 Dec 29 14:41 gameshell
-rwxrwxr-x  1 kali kali  230014 Dec 29 14:41 gameshell-save.sh
-rw-rw-r--  1 kali kali  209794 Jun 17  2025 gameshell.sh
-rw-rw-r--  1 kali kali      55 Dec  3 18:07 .gitconfig
drwx------  3 kali kali    4096 Dec  2 09:25 .gnupg
drwxrwxr-x  4 kali kali    4096 Jan  8 18:54 go
-rw-------  1 kali kali       0 Dec  2 09:25 .ICEauthority
-rw-rw-r--  1 kali kali   63153 Dec  9 17:15 index.html
drwxr-xr-x  4 kali kali    4096 Dec  9 17:22 .java
drwx------  2 kali kali    4096 Jan 15 12:02 .john
drwxr-xr-x  5 kali kali    4096 Dec  2 09:25 .local
-rw-rw-r--  1 kali kali     181 Jan 12 17:05 malware.php
drwx------  4 kali kali    4096 Dec  2 09:28 .mozilla
drwxrwxr-x 12 kali kali    4096 Jan 21 16:12 .msf4
drwxr-xr-x  2 kali kali    4096 Dec 23 18:03 Music
-rw-rw-r--  1 kali kali      29 Dec  9 12:41 output.txt
drwxrwxr-x  3 kali kali    4096 Feb 10 15:20 permessi_lab
```

```
┌──(kali㉿kali)-[~/permessi_lab/shared_reports]
└─$ cat > deploy.sh <<'EOF'
heredoc> #!/bin/bash
echo "Deploy simulato: OK"
EOF

┌──(kali㉿kali)-[~/permessi_lab/shared_reports]
└─$ ls -la
total 16
drwxrwxr-x 2 kali kali 4096 Feb 10 15:36 .
drwxrwxr-x 3 kali kali 4096 Feb 10 15:20 ..
-rw-rw-r-- 1 kali kali   39 Feb 10 15:36 deploy.sh
-rw-rw-r-- 1 kali kali   20 Feb 10 15:30 report.txt
```

2) Screenshot: verifica permessi attuali (prima)

ls -l mostra permessi dei file (report.txt, deploy.sh)

ls -ld mostra permessi della directory shared_reports (importante!)

```
┌──(kali㉿kali)-[~/permessi_lab]
└─$ ls -l shared_reports
total 8
-rw-rw-r-- 1 kali kali 27 Feb 10 15:39 deploy.sh
-rw-rw-r-- 1 kali kali 20 Feb 10 15:30 report.txt

┌──(kali㉿kali)-[~/permessi_lab]
└─$ ls -ld shared_reports
drwxrwxr-x 2 kali kali 4096 Feb 10 15:39 shared_reports
```

3) Screenshot: modifica permessi (chmod) + verifica dopo

Impostazione di un gruppo della directory e dei file

sudo groupadd devteam
Cosa fa: Crea un nuovo "club" chiamato devteam nel sistema.

- In Linux, ogni utente appartiene a un gruppo primario, ma può essere aggiunto a molti gruppi secondari.

sudo usermod -aG devteam "$USER"
Cosa fa: Aggiunge te stesso ($USER) a quel club.

- -a sta per *append* (aggiungi).

- -G sta per *Groups*.
- **Perché si fa?** Così si diventa ufficialmente un membro del devteam. Quando imposteremo i permessi per il "gruppo" sulla cartella, il sistema controllerà se l'utente che prova ad accedere fa parte di devteam.

```
┌──(kali㉿kali)-[~/permessi_lab]
└─$ sudo groupadd devteam
[sudo] password for kali:

┌──(kali㉿kali)-[~/permessi_lab]
└─$ sudo usermod -aG devteam "$USER"
```

sudo chgrp -R devteam shared_reports
Cosa fa: Cambia il "proprietario di gruppo" della cartella e di tutto ciò che c'è dentro.

- chgrp = *Change Group*.
- -R = *Recursive* (applica a cartella + file interni).
- **Risultato:** Prima la cartella apparteneva al gruppo kali. Ora appartiene al gruppo devteam.

```
┌──(kali㉿kali)-[~/permessi_lab]
└─$ ls -l
total 4
drwxrwxr-x 2 kali devteam 4096 Feb 10 15:39 shared_reports
```

Come Leggere i permessi
Le tre sezioni dei permessi (rwx) si leggono così:

1. **Proprietario (User - kali):**
2. **Gruppo (Group - devteam):** Cosa possono fare **tutti i membri** del gruppo devteam (incluso kali, se aggiunto).
3. **Altri (Others):** Cosa può fare il resto del mondo (come l'utente bob).

In Linux, ogni permesso ha un valore:

- **4** = Lettura (r)
- **2** = Scrittura (w)
- **1** = Esecuzione (x)
- **0** = Nessun permesso (-)

• **Tipo**	• **Proprietario (User)**	• **Gruppo (Group)**	• **Altri (Others)**
• d	• rwx	• rwx	• r-x

Applica i permessi alla cartella, Impostiamo: 750 → rwxr-x---

chmod 750 shared_reports

Directory: owner full, group solo read+execute, others niente

- **7** (4+2+1): **Proprietario (Kali)** → Può fare tutto (rwx).
- **5** (4+0+1): **Gruppo (devteam)** → Può leggere ed entrare (r-x), ma non scrivere.
- **0** (0+0+0): **Altri** → Non possono fare assolutamente nulla (---).

Verifica dei permessi cambiati sui file

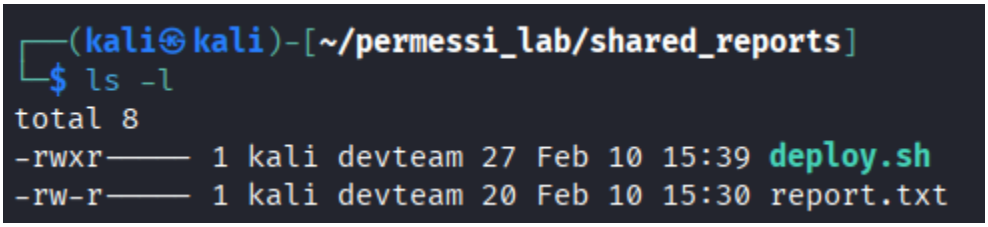
```
┌──(kali㉿kali)-[~/permessi_lab]
└─$ ls -l
total 4
drwxr-x--- 2 kali devteam 4096 Feb 10 15:39 shared_reports
```

Applica i permessi ai file dentro la cartella

chmod 740 shared_reports/deploy.sh
Script: owner eseguibile, group leggibile (ma non eseguibile), others niente

chmod 640 shared_reports/report.txt
File report: owner read/write, group read, others niente

Verifica dei permessi cambiati sui file



---

4) Screenshot: test dei permessi (tentativi + output)

Qui tserve un secondo utente per testare “come vede il team” e “come vede un estraneo”.

Creazione 2 utenti di test (solo lab)

sudo useradd -m alice # Crea l'utente alice

sudo useradd -m bob # Crea l'utente bob

sudo usermod -aG devteam alice # Aggiunge alice al gruppo devteam

Alice (Il Collaboratore Fidato): Fa parte del gruppo devteam. Il sistema riconosce che lei è "una di noi", quindi le permette di leggere i file (permesso r), ma non di modificarli.

Bob (L'Estraneo): Non è nel gruppo. Per lui valgono gli ultimi tre trattini dei permessi (quelli di others). Bob vedrà --- (nessun accesso).

Test come “alice” (gruppo devteam)

```
sudo -iu alice
cd /home/tuo_utente/permessi_lab/shared_reports
ls -l
cat report.txt
echo "aggiunta" >> report.txt
touch nuovo.txt
exit
```

L'errore riscontratoè un classico nel mondo

Alice non riesce a raggiungere la cartella perché la porta principale è chiusa.Perché succede?

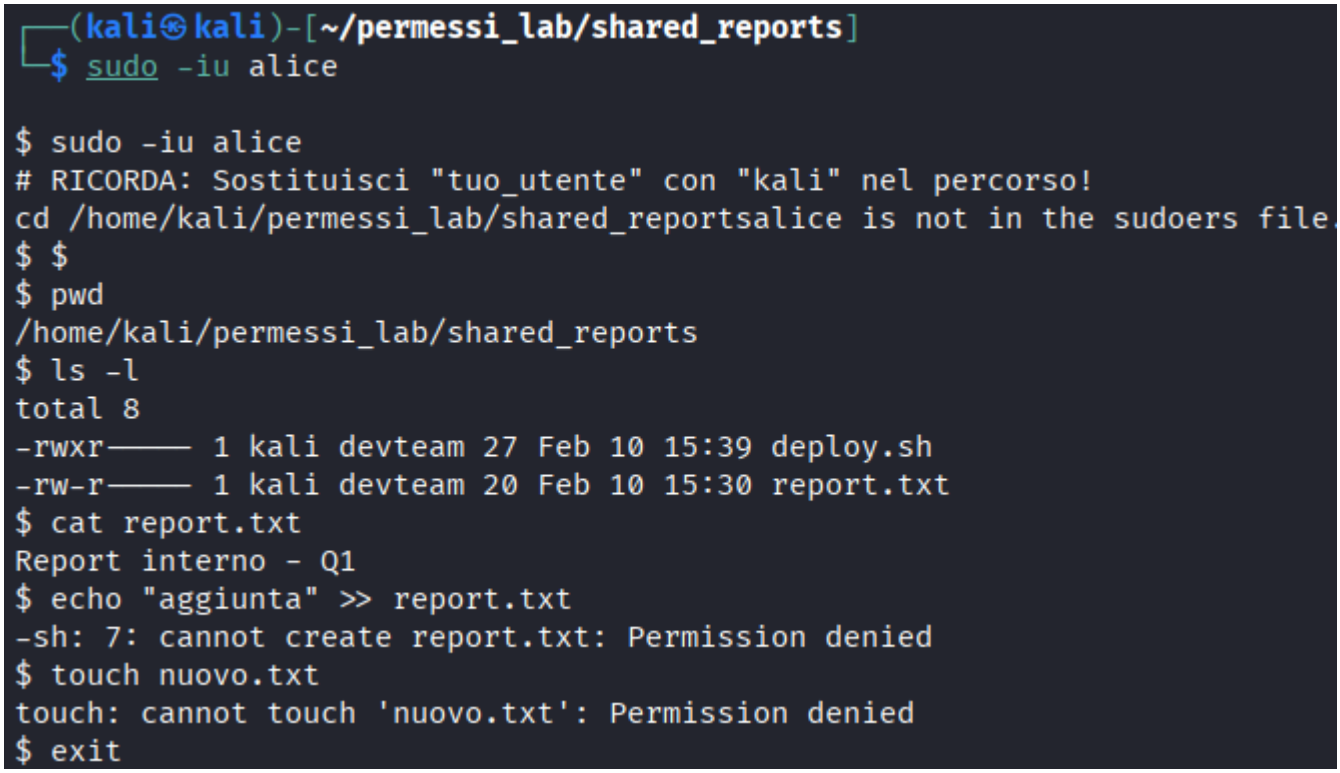
In Linux, per arrivare a una sottocartella (es. shared_reports), un utente deve avere il permesso di "attraversamento" (il bit x) su tutte le cartelle superiori della catena.

Se la tua home directory (/home/kali) è impostata con permessi restrittivi (molto comune su Kali)

Dire al sistema che gli altri utenti possono "attraversare" la cartella home (senza però poterne vedere i file). Uscire da Alice (exit).

chmod o+x /home/kali

Non permette ad Alice di vedere i file personali di kali, ma le permette di "camminare" attraverso la cartella per raggiungere la sottocartella del lab



- cat report.txt OK (perché group ha read)
- echo "aggiunta" >> report.txt FAIL (perché group non ha write sul file)
- touch nuovo.txt FAIL (perché group non ha write sulla directory)

CONCLUSIONE

Scenario

Ho simulato una cartella di progetto “shared_reports” che contiene un report (report.txt) e uno script (deploy.sh). L’obiettivo è consentire al proprietario pieno controllo, al gruppo di lavoro sola consultazione, e impedire l’accesso ad utenti esterni.

Motivazione delle scelte dei permessi

- **Directory shared_reports impostata a 750 (rwxr-x---)**
    - Proprietario: rwx per gestire contenuti e struttura.
    - Gruppo: r-x per poter entrare nella directory e consultare i file, ma senza creare/modificare/cancellare contenuti (manca w).
    - Altri: --- per impedire visibilità e accesso ai contenuti.
- **File report.txt impostato a 640 (rw-r-----)**
    - Proprietario: lettura e scrittura per aggiornare il report.
    - Gruppo: sola lettura per consultazione.
    - Altri: nessun accesso.
- **File deploy.sh impostato a 740 (rwxr-----)**
    - Proprietario: può eseguire lo script (permesso x).
    - Gruppo: può leggerlo per revisione ma non eseguirlo (manca x) per evitare esecuzioni non autorizzate.

- Altri: nessun accesso.

Analisi dei risultati dei test

- L’utente nel gruppo (alice) è riuscito a:
  - entrare nella directory e listare i file (permesso x sulla directory + r).
  - leggere report.txt (permesso r sul file).
- L’utente nel gruppo (alice) **non** è riuscito a:
  - scrivere in report.txt (manca w sul file per il gruppo).
  - creare un nuovo file nella directory (manca w sulla directory per il gruppo).

Conclusione

La configurazione applicata rispetta il principio del **least privilege**: ogni categoria di utente ottiene solo i permessi strettamente necessari. I test hanno confermato che lettura/scrittura/esecuzione e accesso in directory dipendono in modo distinto dai permessi su file e directory