

Build Week 3 - ESERCIZIO 3

Titolo: Navigare nel Filesystem Linux e Impostazioni dei Permessi

Executive Summary

In questo laboratorio è stato esplorato il filesystem Linux e la gestione dei permessi su file e directory, imparando come identificare dispositivi a blocchi, verificare i filesystem montati, montare/smontare partizioni e interpretare permessi/ownership. È stato analizzato i tipi di file Linux (regolari, directory, speciali) e creazione link simbolici e hard link, osservando le differenze operative quando cambiano i nomi dei file originali. L'attività consolida competenze fondamentali di amministrazione e troubleshooting in ambiente Linux, utili anche in ambito security (hardening, triage, incident response).

Introduzione

L'obiettivo dell'esercizio è prendere confidenza con la struttura del filesystem Linux, con i concetti di mount point, permessi (r/w/x), proprietà (owner/group) e con elementi spesso coinvolti in problemi reali di sistema (permessi errati, file speciali, link).

Obiettivi

- Parte 1: Esplorare i filesystem in Linux
 - Parte 2: Permessi dei file
 - Parte 3: Link simbolici e altri tipi di file speciali
-

PARTE 1 — Esplorare i Filesystem in Linux

Passo 1 — Accedere alla riga di comando

1. Avvio la VM **CyberOps Workstation**.
 2. Apro una finestra di **terminale**.
- Terminale aperto con prompt visibile (utente **analyst**).
-

Passo 2 — Visualizzare i filesystem attualmente montati

(a) Visualizzare i dispositivi a blocchi

1. Nel terminale eseguo:
 - `lsblk`
2. Osservo dischi/partizioni e l'eventuale colonna **MOUNTPOINT** (es. `/` su `sda1`).

```
[analyst@secops ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   10G  0 disk 
└─sda1   8:1    0   10G  0 part /
sdb      8:16   0    1G  0 disk 
└─sdb1   8:17   0 1023M 0 part 
sr0     11:0   1 1024M 0 rom
```

- Output completo di `lsblk` con `sda/sda1` e `sdb/sdb1`
-

(b) Visualizzare i filesystem montati

1. Eseguo:
 - `mount`
2. Individuo la riga del filesystem root (es. `/dev/sda1` on `/` type `ext4`).

```
[analyst@secops ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=995444k,nr_inodes=248861,mode=755,inode64)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime)
```

- Output `mount` dove si vede chiaramente `/dev/sda1` on `/` type `ext4`
-

(c) Filtrare solo il filesystem root con grep

1. Eseguo:
 - `mount | grep sda1`

2. Verifico che l'output mostri solo la riga del root filesystem.
3. Mount serve per collegare un file system ad una directory.

```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime)
```

- Output `mount | grep sda1`.
-

(d) Entrare in “/” ed elencare

1. Eseguo in sequenza:
 - `cd /`
 - `ls -l`

DOMANDA (qui): Qual è il significato dell'output? Dove sono fisicamente memorizzati i file elencati?

RISPOSTA: `ls -l` mostra l'elenco dettagliato dei contenuti di `/` (permessi, owner, group, size, data, nome). I file/directory elencati sono **fisicamente memorizzati nel filesystem root**, cioè nella partizione montata su `/` (nell'esempio: `/dev/sda1`).

DOMANDA (qui): Perché `/dev/sdb1` non viene mostrato nell'output sopra?

RISPOSTA: perché `/dev/sdb1` è una **partizione separata** e, se non è montata su un mountpoint, i suoi contenuti non compaiono in `/`. Finché non la monto, è “non visibile” come filesystem navigabile.

```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx  1 root root    7 May  3  2025 bin  -> usr/bin
drwxr-xr-x  3 root root  4096 Jun 18  2025 boot
drwxr-xr-x 20 root root  3920 Feb 23 08:14 dev
drwxr-xr-x 73 root root  4096 Jun 19  2025 etc
drwxr-xr-x  3 root root  4096 Mar 20  2018 home
lrwxrwxrwx  1 root root    7 May  3  2025 lib  -> usr/lib
lrwxrwxrwx  1 root root    7 May  3  2025 lib64 -> usr/lib
drwxr----- 2 root root 16384 Mar 20  2018 lost+found
drwxr-xr-x  2 root root  4096 Jan  5  2018 mnt
drwxr-xr-x  3 root root  4096 Jun 17  2025 opt
dr-xr-xr-x 201 root root     0 Feb 23 08:14 proc
drwxr-x---  8 root root  4096 Jun 18  2025 root
drwxr-xr-x 22 root root   580 Feb 23 08:14 run
lrwxrwxrwx  1 root root    7 May  3  2025 sbin -> usr/bin
drwxr-xr-x  6 root root  4096 Mar 24  2018 srv
dr-xr-xr-x 13 root root     0 Feb 23 08:14 sys
drwxrwxrwt 11 root root  260 Feb 23 09:01 tmp
drwxr-xr-x 10 root root  4096 Jun 19  2025 usr
drwxr-xr-x 12 root root  4096 Jun 19  2025 var
[analyst@secOps /]$
```

- Output `ls -l` in `/` (si deve vedere il prompt che indica directory `/` e la lista).

Passo 3 — Montare e smontare manualmente i filesystem

(a) Verifica directory second_drive

1. Torno nella home (se serve):
 - `cd ~`
2. Controllo la directory:
 - `ls -l second_drive/`
3. Verifico che risulti vuota (`total 0`).

```
[analyst@secOps ~]$ cd ~
[analyst@secOps ~]$ ls -l second_drive/
total 0
[analyst@secOps ~]$
```

- Output `ls -l second_drive/` con `total 0`.
-

(c) Montare /dev/sdb1

1. Eseguo:
 - `sudo mount /dev/sdb1 ~/second_drive/`
2. Se richiesto, inserisco la password.

(d) Elenicare di nuovo il contenuto

1. Eseguo:
 - `ls -l second_drive/`

DOMANDA (qui): Perché la directory non è più vuota? Dove sono fisicamente memorizzati i file elencati?

RISPOSTA: perché ora `second_drive` è diventato il **mount point** di `/dev/sdb1`, quindi mostra i file contenuti nella partizione (`lost+found`, `myFile.txt`, ecc.). I file sono **fisicamente memorizzati su /dev/sdb1**, non “creati” nella directory della home.

📸 Screenshot da fare (Parte 1 / Passo 3c–3d)

- Comando `sudo mount ... oppure` subito dopo lo screen di `ls -l second_drive/` con `lost+found` e `myFile.txt`.
-

(e) Verificare mount dei /dev/sdX

1. Eseguo:
 - `mount | grep /dev/sd`

- Verifico che compaia anche la riga di `/dev/sdb1` montata su `/home/analyst/second_drive`.
Progetto - BW3

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/  
[sudo] password for analyst:  
[analyst@secOps ~]$ ls -l second_drive/  
total 20  
drwx----- 2 root      root     16384 Mar 26  2018 lost+found  
-rw-r--r--  1 analyst    analyst   183 Mar 26  2018 myFile.txt  
[analyst@secOps ~]$
```

- Output `mount | grep /dev/sd`.
-

(f) Smontare `/dev/sdb1` e verificare

- Mi assicuro di essere fuori dal mountpoint:
 - `cd ~`
- Smonto:
 - `sudo umount /dev/sdb1`
- Riconrollo:
 - `ls -l second_drive/` (torna vuota).

```
[analyst@secOps ~]$ cd ~  
[analyst@secOps ~]$ sudo umount /dev/sdb1  
[analyst@secOps ~]$ ls -l second_drive/  
total 0
```

- Output che mostra `sudo umount /dev/sdb1` e/o `ls -l second_drive/` con `total 0`.
-

PARTE 2 — Permessi dei File

Passo 1 — Visualizzare e modificare i permessi dei file

(a) Entrare nella directory scripts

- Eseguo:
 - `cd /home/analyst/lab.support.files/scripts/`
 - Prompt che mostra il path `.../scripts`.
-

(b) Visualizzare permessi con `ls -l`

- Eseguo:
 - `ls -l`

DOMANDA (qui): Considera il file `cyops.mn` come esempio. Chi è il proprietario del file? E il gruppo?

RISPOSTA: Io leggo dalle colonne **owner** e **group** di `ls -l`. Nell'esempio del PDF: owner = `analyst`, group = `analyst`. (Nel report riporto i valori che vedo nella mia VM.)

DOMANDA (qui): I permessi per `cyops.mn` sono `-rw-r--r--`. Cosa significa?

RISPOSTA:

- `-` = file regolare
- `rw-` (owner) = lettura e scrittura
- `r--` (group) = sola lettura
- `r--` (others) = sola lettura

Nessun permesso di esecuzione.

```
[analyst@secOps ~]$ cd /home/analyst/lab.support.files/scripts/
[analyst@secOps scripts]$ ls -l
total 68
-rwxr-xr-x 1 analyst analyst 952 Mar 21 2018 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 1153 Mar 21 2018 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 4053 Jun 18 2025 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 5016 Jun 18 2025 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 4189 Jun 18 2025 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21 2018 cyops.mn
-rwxr-xr-x 1 analyst analyst 458 Mar 21 2018 fw_rules
-rwxr-xr-x 1 analyst analyst 70 Mar 21 2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 net_configuration_files
-rwxr-xr-x 1 analyst analyst 65 Mar 21 2018 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst 189 Mar 21 2018 start_ELK.sh
-rwxr-xr-x 1 analyst analyst 86 Jun 18 2025 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst 86 Jun 19 2025 start_pox.sh
-rwxr-xr-x 1 analyst analyst 117 Jun 19 2025 start_snort.sh
-rwxr-xr-x 1 analyst analyst 61 Mar 21 2018 start_tftpd.sh
```

- Output `ls -l` dove si vede chiaramente la riga `cyops.mn`.

(c) Test permessi: creazione file in /mnt

1. Provo:
 - `touch /mnt/myNewFile.txt`
2. Osservo l'errore "Permission denied".
3. Eseguo:
 - `ls -ld /mnt`
4. Eseguo anche:
 - `ls -ld /` (permessi directory genitore/root).

DOMANDA (qui): Perché il file non è stato creato? Elenca i permessi, la proprietà e il contenuto della directory `/mnt` e spiega cosa è successo. Con l'aggiunta dell'opzione `-d`, elenca i permessi della directory genitore.

RISPOSTA: `/mnt` è di proprietà **root:root** e non è scrivibile dall'utente **analyst** (nell'esempio: `drwxr-xr-x root root ... /mnt`). Senza permesso di **scrittura sulla directory**, non posso creare file al suo interno, quindi `touch` fallisce. Con `ls -ld /` verifico i permessi della directory genitore (root filesystem), tipicamente gestita da root e non scrivibile dagli utenti non privilegiati.

DOMANDA (qui): Cosa si può fare affinché il comando `touch` mostrato sopra abbia successo?

RISPOSTA: posso:

- eseguire il comando con privilegi: `sudo touch /mnt/myNewFile.txt`, oppure
- modificare permessi/ownership di `/mnt` (solo se consentito dalle policy), oppure
- creare il file in una directory dove ho permesso di scrittura (es. `~/` o `/tmp`).

```
[analyst@secOps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
[analyst@secOps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
[analyst@secOps scripts]$ ls -ld /
drwxr-xr-x 17 root root 4096 Jun 19  2025 /
```

- Schermata con:
 - `touch /mnt/myNewFile.txt` e l'errore,
 - `ls -ld /mnt` (permessi/owner/group).
- (Opzionale ma consigliato) un secondo screen con `ls -ld /`.

(d-f) Montare sdb1, leggere permessi di myFile.txt e cambiare con chmod

1. Monto di nuovo:
 - `sudo mount /dev/sdb1 ~/second_drive/`
2. Entro nella directory:
 - `cd ~/second_drive`
3. Elenco:
 - `ls -l`

DOMANDA (qui): Quali sono i permessi del file `myFile.txt`?

RISPOSTA: li leggo dalla prima colonna di `ls -l` sulla riga `myFile.txt`. Nell'esempio del PDF è `-rw-r--r--` (nel report riporto esattamente ciò che vedo).

4. Cambio permessi:
 - `sudo chmod 665 myFile.txt`
5. Ricontrollo:

- o `ls -l`

DOMANDA (qui): I permessi sono cambiati? Quali sono i permessi di `myFile.txt`?

RISPOSTA: sì, dopo `chmod 665` i permessi diventano `-rw-rw-r-x` (come nell'esempio). Nel report riporto la stringa effettiva dell'output.

```
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps scripts]$ cd ~/second_drive
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root      root    16384 Mar 26  2018 lost+found
-rw-r--r--  1 analyst   analyst   183 Mar 26  2018 myFile.txt
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[sudo] password for analyst:
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root      root    16384 Mar 26  2018 lost+found
-rw-rw-r-x  1 analyst   analyst   183 Mar 26  2018 myFile.txt
```

- Screen con `ls -l` prima (permessi iniziali di `myFile.txt`).
 - Screen con `sudo chmod 665 myFile.txt + ls -l` dopo (permessi aggiornati).
-

PARTE 3 — Link simbolici e altri tipi di file speciali

Passo 1 — Esaminare i tipi di file

(a) Tipi di file nella home

1. Eseguo:
 - o `cd ~`
 - o `ls -l`
2. Osservo il primo carattere di ogni riga: `-` (file) o `d` (directory).

```
[analyst@secOps second_drive]$ cd ~
[analyst@secOps ~]$ ls -l
total 44
-rw-r--r-- 1 root      root      24 Feb 17 10:11 capture2.pcap
-rw-r--r-- 1 root      root      0 Feb 17 10:33 capture3.pcap
-rw-r--r-- 1 root      root     5113 Feb 17 10:27 capture.pcap
drwxr-xr-x  3 analyst   analyst  4096 Feb 17 09:25 Desktop
drwxr-xr-x  3 analyst   analyst  4096 Jun 18  2025 Downloads
drwxr-xr-x  9 analyst   analyst  4096 Jun 18  2025 lab.support.files
drwxr-xr-x  3 analyst   analyst  4096 Jun 18  2025 scripts
drwxr-xr-x  3 root      root     4096 Mar 26  2018 second_drive
-rw-r--r--  1 analyst   analyst  314 Feb 19  08:59 space.txt
-rw-r--r--  1 analyst   analyst  313 Feb 19  08:53 spece.txt
drwxr-xr-x  5 analyst   analyst  4096 Jun 18  2025 yay
```

- Output `ls -l` in `/home/analyst` con alcune directory e file visibili.
-

(b) Tipi di file in /dev

1. Eseguo:
 - o `ls -l /dev/`
2. Scorro l'output per individuare:
 - o `b` (block device),
 - o `c` (character device),
 - o `l` (link simbolico).

```
lrwxrwxrwx 1 root root      4 Feb 23 08:14 rtc -> rtc0
crw----- 1 root root    250,   0 Feb 23 08:14 rtc0
brw-rw--- 1 root disk     8,   0 Feb 23 08:14 sda
brw-rw--- 1 root disk     8,   1 Feb 23 08:14 sda1
brw-rw--- 1 root disk     8,  16 Feb 23 08:14 sdb
brw-rw--- 1 root disk     8,  17 Feb 23 08:14 sdb1
drwxrwxrwt 2 root root    40 Feb 23 08:14 shm
crw----- 1 root root    10, 231 Feb 23 08:14 snapshot
drwxr-xr-x 3 root root    180 Feb 23 08:14 snd
brw-rw----+ 1 root optical 11,   0 Feb 23 08:14 sr0
lrwxrwxrwx 1 root root    15 Feb 23 08:14 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root    15 Feb 23 08:14 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root    15 Feb 23 08:14 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty      5,   0 Feb 23 09:52 tty
```

- Porzione dell'elenco `/dev` dove si vedono chiaramente righe che iniziano con `b`, `c` e `l` (es. `sda`, `tty`, `rtc -> rtc0`).

(c-f) Creare file, creare link simbolico e hard link, e testare cosa succede rinominando

(c) Creare due file con echo

1. Eseguo:
 - o `echo "symbolic" > file1.txt`
 - o `cat file1.txt`
2. Eseguo:
 - o `echo "hard" > file2.txt`
 - o `cat file2.txt`

```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$ echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
```

- Comandi `echo` e output `cat` di entrambi i file.

(d) Creare un link simbolico e un hard link

1. Creo link simbolico:
 - o `ln -s file1.txt file1symbolic`
2. Creo hard link:
 - o `ln file2.txt file2hard`

```
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
[analyst@secOps ~]$
```

- Comandi `ln -s ...` e `ln ...` eseguiti (anche senza output).
-

(e) Verificare con ls -l e interpretare

1. Eseguo:
 - o `ls -l`
2. Verifico che:
 - o `file1symbolic` inizi con `l` e mostri `-> file1.txt`
 - o `file2hard` appaia come file regolare ma con link count `2` (quinta colonna), come anche `file2.txt`.

```
[analyst@secOps ~]$ ls -l
total 56
-rw-r--r-- 1 root      root      24 Feb 17 10:11 capture2.pcap
-rw-r--r-- 1 root      root      0 Feb 17 10:33 capture3.pcap
-rw-r--r-- 1 root      root     5113 Feb 17 10:27 capture.pcap
drwxr-xr-x 3 analyst  analyst  4096 Feb 17 09:25 Desktop
drwxr-xr-x 3 analyst  analyst  4096 Jun 18 2025 Downloads
lrwxrwxrwx 1 analyst  analyst   9 Feb 23 10:22 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst  analyst   9 Feb 23 10:19 file1.txt
-rw-r--r-- 2 analyst  analyst   5 Feb 23 10:20 file2hard
-rw-r--r-- 2 analyst  analyst   5 Feb 23 10:20 file2.txt
drwxr-xr-x 9 analyst  analyst  4096 Jun 18 2025 lab.support.files
drwxr-xr-x 3 analyst  analyst  4096 Jun 18 2025 scripts
drwxr-xr-x 3 root      root    4096 Mar 26 2018 second_drive
-rw-r--r-- 1 analyst  analyst  314 Feb 19 08:59 space.txt
-rw-r--r-- 1 analyst  analyst  313 Feb 19 08:53 spece.txt
drwxr-xr-x 5 analyst  analyst  4096 Jun 18 2025 yay
```

- Output `ls -l` dove si vedono `file1symbolic -> file1.txt` e i "2" in quinta colonna per `file2hard` e `file2.txt`.
-

(f) Rinominare i file originali e osservare l'impatto

1. Rinomino:
 - o `mv file1.txt file1new.txt`
 - o `mv file2.txt file2new.txt`
2. Provo:

- `cat file1symbolic` (deve risultare “rotto”/errore)
3. Provo:
- `cat file2hard` (deve funzionare e mostrare `hard`)

DOMANDA (qui): Cosa pensi succederebbe a `file2hard` se aprissi un editor di testo e cambiassi il testo in `file2new.txt`?

RISPOSTA: `file2hard` e `file2new.txt` puntano allo stesso inode (stesso contenuto fisico). Se modifico il contenuto in `file2new.txt`, anche `file2hard` mostrerà il **contenuto aggiornato**, perché sono due “nomi” per lo stesso file a livello di filesystem.

```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
[analyst@secOps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@secOps ~]$ cat file2hard
hard
```

- Screen con i comandi `mv ...`, `cat file1symbolic` (errore) e `cat file2hard` (output “hard”).
-

Conclusione

L'esercizio ha permesso di consolidare competenze essenziali su Linux: identificazione dispositivi e filesystem, uso corretto di mount/umount, interpretazione di permessi e proprietà (cause frequenti di errori operativi), e comprensione dei diversi tipi di file e dei meccanismi di linking (symlink vs hard link).

Queste basi sono cruciali sia per l'amministrazione di sistema sia per attività di security (hardening, verifica accessi, analisi incidenti e troubleshooting)