Build Week 3

Esercizio 3

LANDA

TRACKER SPA

In questo laboratorio, prenderai familiarità con i filesystem Linux.

- Parte 1 Esplorare i Filesystem in Linux
- Parte 2 Permessi dei File
- Parte 3 Link Simbolici e Altri Tipi di File Speciali

PARTE 1: FILESYSTEM IN LINUX

Qual è il significato dell'output?

L'output ritorna come risultato la directory completa del filesystem root.

```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx   1 root root     7 Jan  5  2018 bin -> usr/bin
drwxr-xr-x   3 root root  4096 Apr 16  2018 boot
drwxr-xr-x  20 root root  3240 Sep 29 06:28 dev
drwxr-xr-x  58 root root  4096 Jan 31  2025 etc
drwxr-xr-x   3 root root  4096 Mar 20  2018 home
lrwxrwxrwx   1 root root     7 Jan  5  2018 lib -> usr/lib
lrwxrwxrwx   1 root root     7 Jan  5  2018 lib64 -> usr/lib
drwx------   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 Jan 31  2025 opt
dr-xr-xr-x 139 root root     0 Sep 29 06:28 proc
drwxr-x---   7 root root  4096 Sep 23 09:19 root
drwxr-xr-x  18 root root   500 Sep 29 06:28 run
lrwxrwxrwx   1 root root     7 Jan  5  2018 sbin -> usr/bin
drwxr-xr-x   6 root root  4096 Mar 24  2018 srv
dr-xr-xr-x  13 root root     0 Sep 29 06:28 sys
drwxrwxrwt   8 root root   200 Sep 29 06:36 tmp
drwxr-xr-x   9 root root  4096 Jan 31  2025 usr
drwxr-xr-x  12 root root  4096 Jan 31  2025 var
```

Dove sono fisicamente memorizzati i file elencati?

I file elencati sono memorizzati sul disco **sda**, all'interno della partizione **sda1**.

Perché /dev/sdb1 non viene mostrato nell'output sopra?

La partizione **sdb1** non viene mostrata perché come possiamo notare anche dal **mountpoint**, questo hard disk non ha alcun mount. Questo significa che il dispositivo non è montato o è vuoto/non formattato.

```
[analyst@secOps /]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
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
```

Perché la directory non è più vuota?

La directory non è più vuota perché abbiamo montato **/dev/sdb1** sulla cartella **second_drive** creata precedentemente

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps ~]$ ls -l
total 24
-rw-r--r-- 1 root    root    6180 Sep 23 08:49 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Mar 22  2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Jan 31  2025 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 19  2018 lab.support.files
drwxr-xr-x 3 root    root    4096 Mar 26  2018 second_drive
[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
```

Dove sono fisicamente memorizzati i file elencati?

I file elencati sono fisicamente montati nel disco sdb nella partizione sdb1. Infatti adesso notiamo il path dal MOUNTPOINT

```
[analyst@secOps ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
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 /home/analyst/second_drive
sr0     11:0    1 1024M  0 rom
```

PARTE 2: PERMESSI DEI FILE

Considera il file cyops.mn come esempio. Chi è il proprietario del file? E il gruppo?

Il file **cyops.mn** è di proprietà di **analyst**, gruppo analyst

```
[analyst@secOps scripts]$ ls -l
total 60
-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 3459 Mar 21  2018 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Mar 21  2018 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Mar 21  2018 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   85 Mar 21  2018 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst   76 Mar 21  2018 start_pox.sh
-rwxr-xr-x 1 analyst analyst  106 Mar 21  2018 start_snort.sh
-rwxr-xr-x 1 analyst analyst   61 Mar 21  2018 start_tftpd.sh
```

I permessi per cyops.mn sono -rw-r--r--. Cosa significa?

I permessi **(-)(rw-)(r--)(r--)** per il file cyops.mn significano che il file ha i seguenti permessi di accesso:

- L'elemento iniziale **-** indica che si tratta di un file.
- La prima tripletta **rw-** indica che il proprietario ha i permessi di lettura e scrittura, ma non di esecuzione.
- La seconda tripletta **r--** indica che i membri del gruppo a cui appartiene il file hanno solo il permesso di lettura.
- La tripletta finale **r--** indica che tutti gli altri utenti hanno solo il permesso di lettura.

```
[analyst@secOps scripts]$ stat cyops.mn
  File: cyops.mn
  Size: 2871            Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 18177       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ analyst)   Gid: ( 1000/ analyst)
Access: 2018-03-21 13:06:40.902145644 -0400
Modify: 2018-03-21 13:06:40.903145644 -0400
Change: 2018-03-21 13:06:40.903145644 -0400
```

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. Registra la risposta nelle righe sottostanti.

Il file non è stato creato perché l'utente analyst non ha i permessi di scrittura per poter creare il file .txt all'interno della directory mnt.

```
[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
```

Cosa si può fare affinché il comando touch mostrato sopra abbia successo?

Per fare in modo che il comando touch abbia successo bisogna dare i permessi all'interno della directory o utilizzare un utente che ha già i permessi necessari per la creazione del file.

Quali sono i permessi del file myFile.txt?

I permessi del file myFile.txt sono **(-)(rw-)(r--)(r--)**. Possiamo tradurli nel seguente modo:

- L'elemento iniziale **-** indica che si tratta di un file normale.
- La prima tripletta **rw-** indica che il proprietario ha i permessi di lettura e scrittura.
- La seconda tripletta **r--** indica che i membri del gruppo a cui appartiene il file hanno solo il permesso di lettura.
- La terza tripletta **r--** indica che tutti gli altri utenti hanno solo il permesso di lettura.

```
[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]$
```

I permessi sono cambiati? Quali sono i permessi di myFile.txt?

Si, i permessi sono cambiati tramite il comando **chmod 665** nel seguente modo: (rw-)(rw-)(r-x).

- La prima tripletta **rw-** indica lettura e scrittura per il proprietario.
- La seconda tripletta **rw-** indica lettura e scrittura per il gruppo.
- La terza tripletta **r-x** indica lettura e esecuzione per tutti gli altri.

```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[analyst@secOps second_drive]$ ls-l
bash: ls-l: command not found
[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
```

Quale comando cambierebbe i permessi di myFile.txt a rwxrwxrwx, garantendo a qualsiasi utente nel sistema pieno accesso al file?

Il comando che cambierebbe i permessi di **myFile.txt** a (**rwx)(rwx)(rwx)**, garantendo a qualsiasi utente nel sistema di leggere, modificare ed eseguire l'elemento è:

- c**hmod 777**

oppure:

- c**hmod u+rwx,g+rwx,o+rwx myFile.txt**

```
[analyst@secOps second_drive]$ sudo chmod 777 myFile.txt
[sudo] password for analyst:
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root    root    16384 Mar 26  2018 lost+found
-rwxrwxrwx 1 analyst analyst   183 Mar 26  2018 myFile.txt
```

L'operazione è riuscita? Spiega.

L'operazione è riuscita. Grazie al cambio di permessi fatto in precedenza siamo stati in grado di scrivere all'interno del file **myFile.txt.**

```
[analyst@secOps second_drive]$ echo Castle Slept >> myFile.txt
[analyst@secOps second_drive]$ cat myFile.txt
This is a file stored in the /dev/sdb1 disk.
Notice that even though this file has been sitting in this disk for a while, it couldn't be accessed until the disk was properly mounted.
test
Castle Slept
```

Qual è la differenza tra la parte iniziale della riga di malware e la riga di mininet_services?

La differenza tra la riga di malware e la riga di **mininet_services** sta nel primo carattere della riga dei permessi: la riga di malware presenta una **d** all'inizio della riga, questo indica che quella è una directory.
Al contrario nel file mininet_services quel valore non c'è, di conseguenza è un file e non una directory.

```
drwxr-xr-x 2 analyst analyst    4096 Mar 21   2018 malware
-rwxr-xr-x 1 analyst analyst     172 Mar 21   2018 mininet_services
```

PARTE 3: LINK SIMBOLICI E ALTRI TIPI DI FILE

Cosa pensi succederebbe a file2hard se aprissi un editor di testo e cambiassi il testo in file2new.txt?

Se cambiassimo il nome di **file2hard** in **file2new**, non succederebbe niente al link che abbiamo creato, poiché gli hard link si collegano allo stesso inode, quindi non sono sensibili alle modifiche dei nomi, al contrario del symbolic link.

```
[analyst@secOps ~]$ echo LASAGNA >> new.txt
[analyst@secOps ~]$ cat new.txt
Sleep While Mark Is Racing With a Cat
LASAGNA
[analyst@secOps ~]$ cat file2Mark
Sleep While Mark Is Racing With a Cat
LASAGNA
```