



# FILE SYSTEM

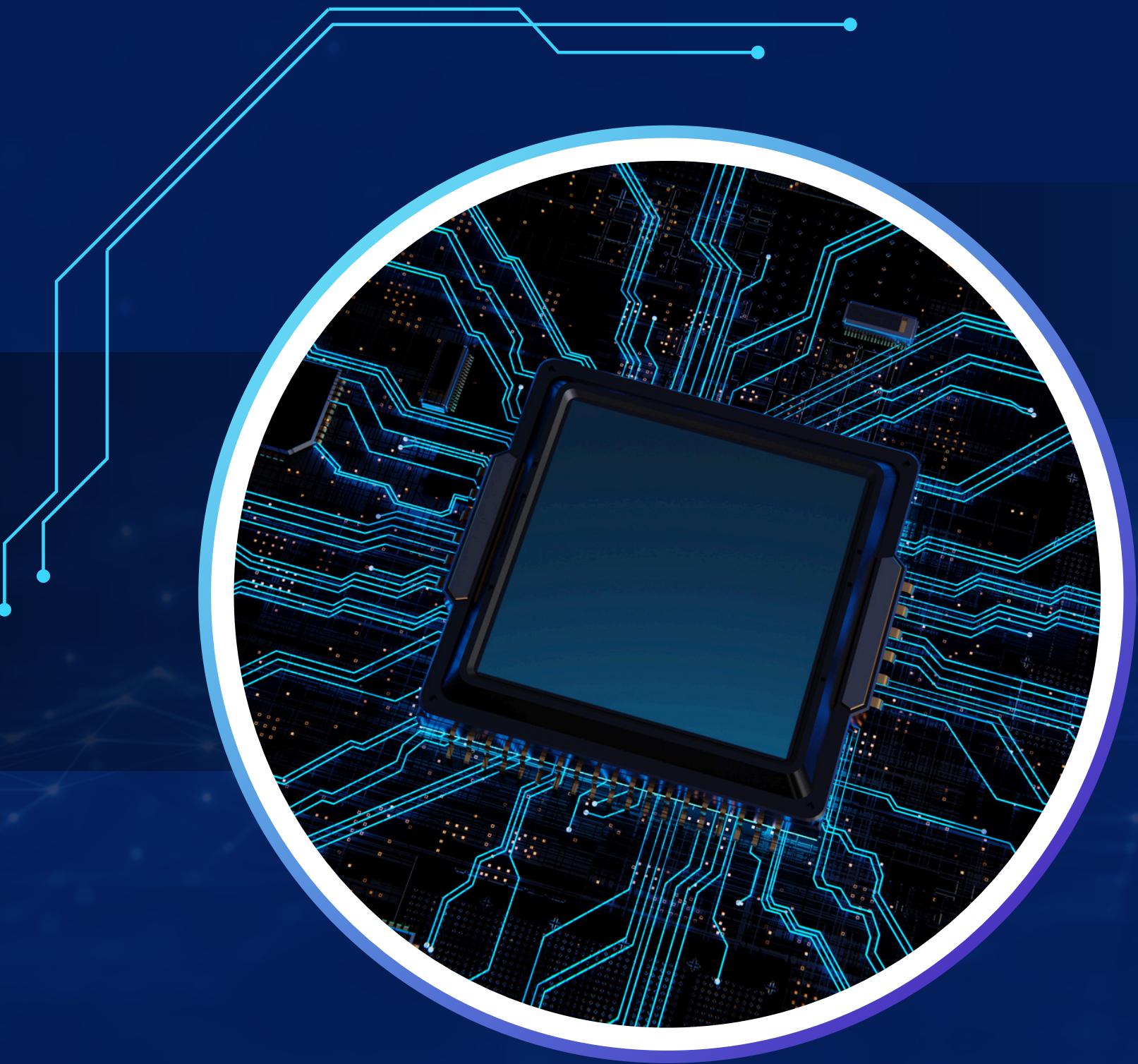
# COSA SONO?

I file system in Linux sono i meccanismi utilizzati per organizzare, archiviare e accedere ai file sui dispositivi di archiviazione come hard disk, SSD, USB, ecc.

Linux supporta vari tipi di file system, ognuno con caratteristiche specifiche per determinati utilizzi. Un file system è una struttura logica che consente di:

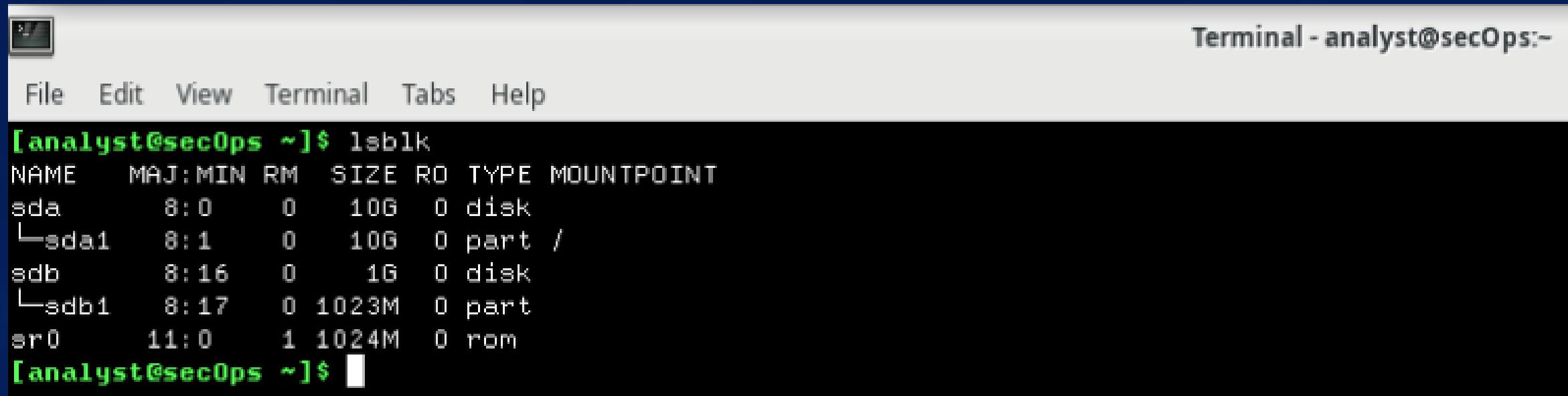
- Organizzare i dati in file e directory.
- Gestire l'accesso ai dati.
- Allocare e liberare spazio sui dispositivi di archiviazione.

Prima di poter essere utilizzati, i file system vanno montati. Montare un filesystem significa renderlo accessibile al sistema operativo. Montare un filesystem è il processo di collegamento della partizione fisica sul dispositivo a blocchi (disco rigido, unità SSD, chiavetta USB, ecc.) a una directory, tramite la quale è possibile accedere all'intero file system. Poiché la suddetta directory diventa la radice del filesystem appena montato, è anche nota come punto di montaggio



## ■ Vision

Utilizzando il comando `lsblk` si possono visualizzare tutti i dispositivi a blocchi:



The screenshot shows a terminal window titled "Terminal - analyst@secOps:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the output of the `lsblk` command. The output is as follows:

```
[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 
[analyst@secOps ~]$
```

L'immagine mostra che la VM ha 3 dispositivi a blocchi: `sda`, `sdb`, `sr0`.

Nei computer con più dischi rigidi si visualizzeranno più dispositivi `/dev/sdX`. Se Linux fosse in esecuzione su un computer con quattro dischi rigidi, ad esempio, li mostrerebbe come `/dev/sda`, `/dev/sdb`, `/dev/sdc` e `/dev/sdd`, per impostazione predefinita. Lo screen implica che `sda` e `sdb` sono dischi rigidi, ognuno contenente una singola partizione.

Lo screen ad albero mostra anche le partizioni sotto `sda` e `sdb`.

## ■ Filesystem root

Per poter visualizzare solo il filesystem root, si esegue sempre il comando mount più grep per filtrare l'output.

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



## ■ Cloud Security

Mount mostra che il filesystem root si trova nella prima partizione del dispositivo a blocchi sda (`/dev/sda1`).

L'output dice anche il tipo di formattazione utilizzato nella partizione, ext4 in questo caso.

Le informazioni tra parentesi si riferiscono alle opzioni di montaggio della partizione.

Il comando mount può anche essere utilizzato per montare e smontare i filesystem.

Come visto, la VM ha due dischi rigidi: il primo è stato riconosciuto dal kernel come `/dev/sda`, mentre il secondo è stato riconosciuto come `/dev/sdb`.



Il comando **mount** serve per visualizzare informazioni più dettagliate sui file system attualmente montati nella VM.

```
[analyst@sec0ps ~]$ 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=500780k,nr_inodes=125195,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=31,pgrp=1,timeo=0,minproto=5,maxproto=5,direct,pipe_ino=10434)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
configfs on /sys/kernel/config type configfs (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=101288k,mode=700,uid=1000,gid=1000)
[analyst@sec0ps ~]$
```

Concentriamoci sul filesystem root , il filesystem è memorizzato in `/dev/sda1` . Il filesystem root è dove è memorizzato il sistema operativo Linux stesso; tutti i programmi, gli strumenti e i file di configurazione sono memorizzati nel filesystem root per impostazione predefinita.

# IL PUNTO DI MONTAGGIO

## ■ mkdir

Prima che un dispositivo a blocchi possa essere montato, deve avere un punto di montaggio.

```
[analyst@secOps scripts]$ mkdir second_drive  
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/  
[sudo] password for analyst:  
[analyst@secOps scripts]$ cd ~/second_drive  
[analyst@secOps second-drive]$ █
```

Lo screen mostra come viene creata la directory second\_drive utilizzando il comando mkdir (usato proprio per creare directory). Con mount, invece, si monta /dev/sdb1 sulla directory appena creata.



# FILESYSTEM

Ora che `/dev/sdb1` è stato montato su `/home/analyst/second_drive`, utilizzo il comando `ls -l` per elencare il contenuto della directory.

```
[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]$
```

## lab.support

Ogni file nel filesystem ha il suo set di permessi che definiscono cosa gli utenti e i gruppi possono fare con il file (lettura, scrittura, esecuzione).

Per modificare le autorizzazioni dei file bisogna recarsi nella directory files: `/home/analyst/lab.support.files/scripts/` e visualizzare la lista di permessi dei file.





# SECURITY TRAINING AND AWARENESS

Consideriamo `cyops.mn` come esempio, il proprietario del file è “`analyst`” così come il gruppo.

I permessi per `cyops.mn` sono `-rw-r-r-`. Ciò significa che il proprietario del file “`analyst`” può leggere e scrivere sul file ma non eseguirlo (`-rw`).

I membri del gruppo “`analyst`”, diversi dal proprietario, possono solo leggere il file (`-r-`), nessuna esecuzione o scrittura è consentita. A tutti gli altri utenti non è consentito scrivere o eseguire quel file.

```
[analyst@sec0ps scripts]$ ls -l
total 60
-rw-r--r-- 1 analyst analyst 952 Mar 21 2018 configure_as_dhcp.sh
-rw-r--r-- 1 analyst analyst 1153 Mar 21 2018 configure_as_static.sh
-rw-r--r-- 1 analyst analyst 3459 Mar 21 2018 cyberops_extended_topo_no_fw.py
-rw-r--r-- 1 analyst analyst 4062 Mar 21 2018 cyberops_extended_topo.py
-rw-r--r-- 1 analyst analyst 3669 Mar 21 2018 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21 2018 cyops.mn
-rw-r--r-- 1 analyst analyst 458 Mar 21 2018 fw_rules
-rw-r--r-- 1 analyst analyst 70 Mar 21 2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 net_configuration_files
-rw-r--r-- 1 analyst analyst 65 Mar 21 2018 reg_server_start.sh
-rw-r--r-- 1 analyst analyst 189 Mar 21 2018 start_EJK.sh
-rw-r--r-- 1 analyst analyst 85 Mar 21 2018 start_miniedit.sh
-rw-r--r-- 1 analyst analyst 76 Mar 21 2018 start_pox.sh
-rw-r--r-- 1 analyst analyst 106 Mar 21 2018 start_snort.sh
-rw-r--r-- 1 analyst analyst 61 Mar 21 2018 start_tftpd.sh
[analyst@sec0ps scripts]$
```

# FILESYSTEM

## Il comando touch

Il comando touch permette la creazione di un file di testo.

Creando un file nella directory /mnt, darà errore.

I permessi della directory /mnt sono di proprietà dell'utente root, con permessi drwxr-xr-x.

In questo modo, solo l'utente root è autorizzato a scrivere nella cartella /mnt.

```
alyst@secOps scripts]$ touch /mnt/myNewFile.txt  
ch: cannot touch '/mnt/myNewFile.txt': Permission denied  
alyst@secOps scripts]$ █
```

Per far sì che il comando abbia successo, deve essere eseguito come root (sudo), oppure sarà possibile modificare i permessi della directory /mnt.

Il comando chmod viene utilizzato per modificare i permessi di un file o di una directory. Come prima c'è, monta la partizione /dev/sdb1 sulla directory /home/analyst/second\_drive, creata in precedenza.

Si usa anche per modificare i permessi di myFile.txt, in questo caso.

```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt  
[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  
[analyst@secOps second_drive]$ █
```

Ora i permessi di myFile.txt sono -rw-rw-rx.

- Proprietario: può leggere e scrivere, ma non esegue;
- Gruppo: possono leggere e scrivere, ma non eseguire;
- Altri utenti: possono leggere ed eseguire, ma non scrivere.

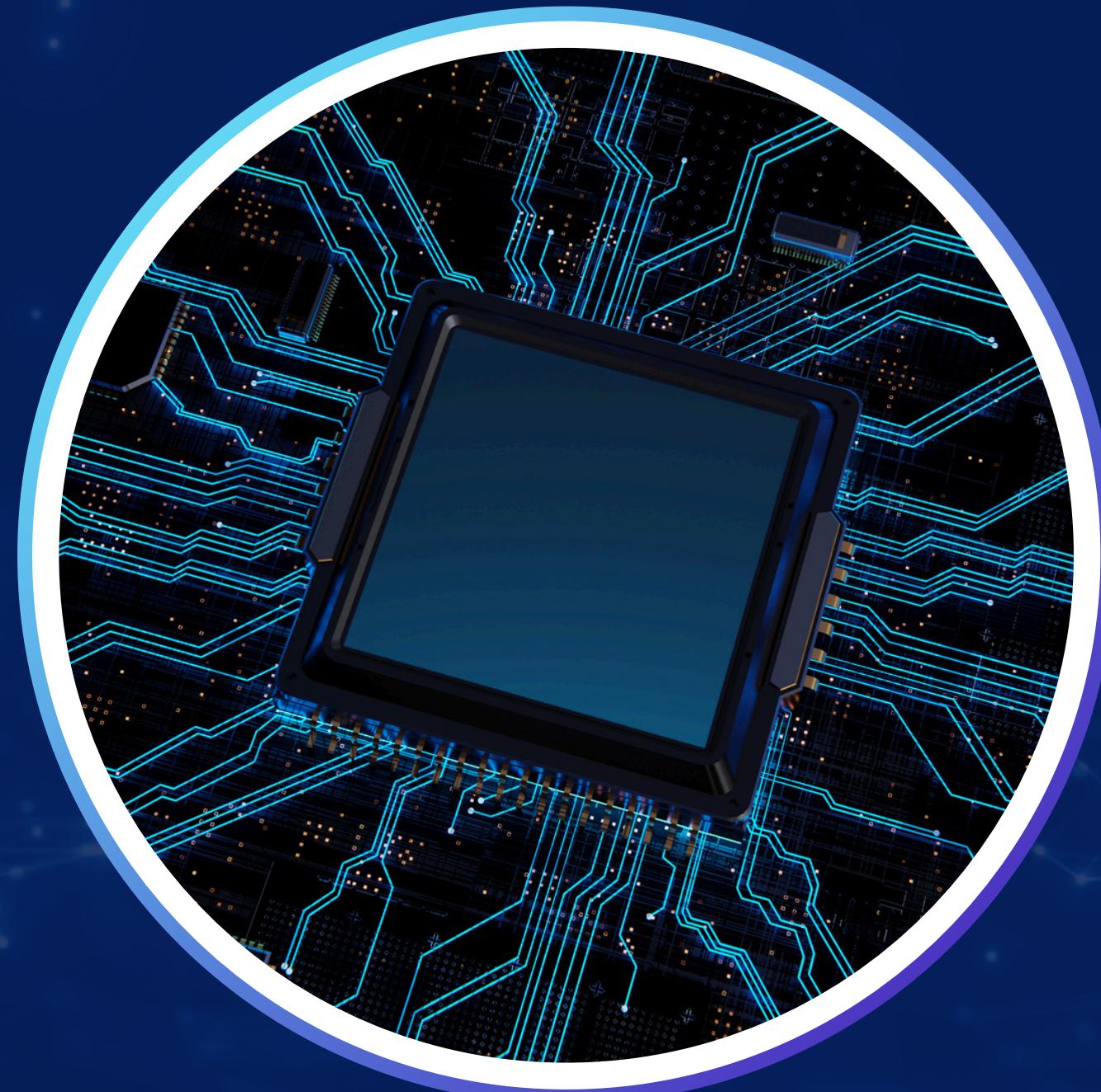
Per garantire a tutti l'accesso completo ai file si dovrebbe eseguire il comando: sudo chmod 777 myFile.txt, comando che cambierebbe i permessi in rwxrwxrwx.

Per rendere root il proprietario di myFile.txt, si usa il comando chown:

```
[analyst@secOps second_drive]$ sudo chown analyst myFile.txt
[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
[analyst@secOps second_drive]$
```

Ora l'analista è proprietario del file.

Anche le directory hanno permessi. Ci sono anche permessi speciali: setuid, setgid e sticky.



Tornando alla directory di inizio (/home/analyst/lab.support.files), si esegue il comando lista per elencare i file dettagliati.

```
[analyst@secOps second_drive]$ cd ~/lab.support.files/
[analyst@secOps lab.support.files]$ ls -l
total 580
-rw-r--r-- 1 analyst analyst      649 Mar 21 2018 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst      126 Mar 21 2018 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst    4096 Mar 21 2018 attack_scripts
-rw-r--r-- 1 analyst analyst      102 Mar 21 2018 confidential.txt
-rw-r--r-- 1 analyst analyst    2871 Mar 21 2018 cyops.mn
-rw-r--r-- 1 analyst analyst       75 Mar 21 2018 elk_services
-rw-r--r-- 1 analyst analyst      373 Mar 21 2018 h2_dropbear.banner
drwxr-xr-x 2 analyst analyst    4096 Apr  2 2018 instructor
-rw-r--r-- 1 analyst analyst      255 Mar 21 2018 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst   24464 Mar 21 2018 logstash-tutorial.log
drwxr-xr-x 2 analyst analyst    4096 Mar 21 2018 malware
-rwrxr-xr-x 1 analyst analyst      172 Mar 21 2018 mininet_services
drwxr-xr-x 2 analyst analyst    4096 Mar 21 2018 openssl_lab
drwxr-xr-x 2 analyst analyst    4096 Mar 21 2018 pcaps
drwxr-xr-x 7 analyst analyst    4096 Mar 21 2018 pox
-rw-r--r-- 1 analyst analyst  473363 Mar 21 2018 sample.img
-rw-r--r-- 1 analyst analyst       65 Mar 21 2018 sample.img_SHA256.sig
drwxr-xr-x 4 analyst analyst    4096 Dec 16 10:39 scripts
-rw-r--r-- 1 analyst analyst  25553 Mar 21 2018 SQL_Lab.pcap
[analyst@secOps lab.support.files]$ █
```

La lettera 'd' all'inizio della riga indica che il tipo di file è una directory e non un file.

Un'altra differenza tra i permessi di file e directory è il bit di esecuzione.

Se un file ha il suo bit di esecuzione attivato, significa che può essere eseguito dal sistema.

Le directory sono diverse dai file con il bit di esecuzione impostato (un file con il bit di esecuzione impostato è uno script o un programma eseguibile).

Una directory con il bit di esecuzione impostato specifica se un utente può entrare in quella directory.

Ora visualizzerò i file e directory in /home/analyst con ls - l.

```
[analyst@secOps ~]$ ls -l
total 4972
-rw-r--r-- 1 root      root      5551 Dec 11 07:02 capture.pcap
drwxr-xr-x  2 analyst   analyst   4096 Mar 22 2018 Desktop
drwxr-xr-x  3 analyst   analyst   4096 Mar 22 2018 Downloads
-rw-r--r-- 1 root      root     5063503 Dec 13 05:09 httpdump.pcap
drwxr-xr-x  9 analyst   analyst   4096 Jul 19 2018 lab.support.files
drwxr-xr-x  2 analyst   analyst   4096 Mar 21 2018 second_drive
[analyst@secOps ~]$
```



I link simbolici sono come le scorciatoie in Windows.

Ci sono due tipi di link: simbolici e hard.

Da notare che i primi caratteri di ogni riga sono o un “-“ che indica un file o una “d” che indica una directory e nota come i file di blocco inizia con una “b”, i file di dispositivo a caratteri iniziano con una “c” e i file di collegamento simbolico iniziano con una “l”:

```
[analyst@secOps ~]$ ls -l /dev/
total 0
crw-r--r--  1 root root      10, 235 Dec 16 09:16 autofs
drwxr-xr-x  2 root root      140 Dec 16 09:16 block
drwxr-xr-x  2 root root      100 Dec 16 09:16 bsg
crw-----  1 root root     10, 234 Dec 16 09:16 btrfs-control
drwxr-xr-x  3 root root      60 Dec 16 09:16 bus
lrxwxrwxrwx 1 root root      3 Dec 16 09:16 cdrom -> sr0
drwxr-xr-x  2 root root     2800 Dec 16 09:16 char
crw-----  1 root root      5,  1 Dec 16 09:16 console
lrxwxrwxrwx 1 root root      11 Dec 16 09:16 core -> /proc/kcore
crw-----  1 root root     10,  61 Dec 16 09:16 cpu_dma_latency
crw-----  1 root root      10, 203 Dec 16 09:16 cuse
drwxr-xr-x  6 root root      120 Dec 16 09:16 disk
drwxr-xr-x  3 root root      80 Dec 16 09:16 dri
crw-rw----  1 root video    29,  0 Dec 16 09:16 fb0
lrxwxrwxrwx 1 root root      13 Dec 16 09:16 fd -> /proc/self/fd
crw-rw-rw-  1 root root      1,  7 Dec 16 09:16 full
crw-rw-rw-  1 root root     10, 229 Dec 16 09:16 fuse
crw-----  1 root root     245,  0 Dec 16 09:16 hidraw0
crw-rw----  1 root audio     10, 228 Dec 16 09:16 hpet
drwxr-xr-x  2 root root      0 Dec 16 09:16 hugepages
lrxwxrwxrwx 1 root root      25 Dec 16 09:16 initctl -> /run/systemd/initctl/fifo
drwxr-xr-x  4 root root     360 Dec 16 09:16 input
crw-r--r--  1 root root      1, 11 Dec 16 09:16 kmsg
drwxr-xr-x  2 root root     60 Dec 16 09:16 lightning
```

La differenza tra link simbolici e hard link è che un file simbolico punta al nome file di un altro file e un file di hard punta al contenuto di un altro file. Vediamoli:

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

- Nota come il file file1symbolic sia un collegamento simbolico con una l all'inizio della riga e un puntatore -> a file1.txt .

- Il file2hard sembra essere un file normale, perché in effetti è un file normale che punta allo stesso inode sul disco rigido come file2.txt.

- Per un elenco di directory, la quinta colonna indica il numero di directory all'interno della directory, incluse le cartelle nascoste.

Ora si eseguirà un collegamento simbolico a file1 e collegamento fisico in file2.

```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$ echo "difficult" > file2.txt
[analyst@secOps ~]$ cat file2.txt
difficult
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2difficult
[analyst@secOps ~]$ ls -l
total 4984
-rw-r--r-- 1 root      root          5551 Dec 11 07:02 capture.pcap
drwxr-xr-x  2 analyst   analyst        4096 Mar 22 2018 Desktop
drwxr-xr-x  3 analyst   analyst        4096 Mar 22 2018 Downloads
lrwxrwxrwx  1 analyst   analyst         9 Dec 16 11:42 file1symbolic -> file1.txt
-rw-r--r--  1 analyst   analyst         9 Dec 16 11:40 file1.txt
-rw-r--r--  2 analyst   analyst        10 Dec 16 11:41 file2difficult
-rw-r--r--  2 analyst   analyst        10 Dec 16 11:41 file2.txt
-rw-r--r--  1 root      root      5063503 Dec 13 05:09 httpdump.pcap
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 ~]$
```

Cambiando il nome dei file originali, i file collegati subiranno un effetto.

```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
[analyst@secOps ~]$ ls -l
total 4984
-rw-r--r-- 1 root      root          5551 Dec 11 07:02 capture.pcap
drwxr-xr-x  2 analyst   analyst       4096 Mar 22 2016 Desktop
drwxr-xr-x  3 analyst   analyst       4096 Mar 22 2018 Downloads
-rw-r--r--  1 analyst   analyst        9 Dec 16 11:40 file1new.txt
lrwxrwxrwx  1 analyst   analyst      9 Dec 16 11:42 file1symbolic -> file1.txt
-rw-r--r--  2 analyst   analyst       10 Dec 16 11:41 file2difficult
-rw-r--r--  2 analyst   analyst       10 Dec 16 11:41 file2new.txt
-rw-r--r--  1 root      root      5063503 Dec 13 05:09 httpdump.pcap
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 ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@secOps ~]$ cat file2difficult
difficult
[analyst@secOps ~]$
```

Si noti che file1symbolic è ora un collegamento simbolico non funzionante perché il nome del file a cui puntava file1.txt è cambiato, ma il file di collegamento fisso file2hard funziona ancora correttamente perché punta all'inode di file2.txt e non al suo nome, che ora è file2new.txt .

