

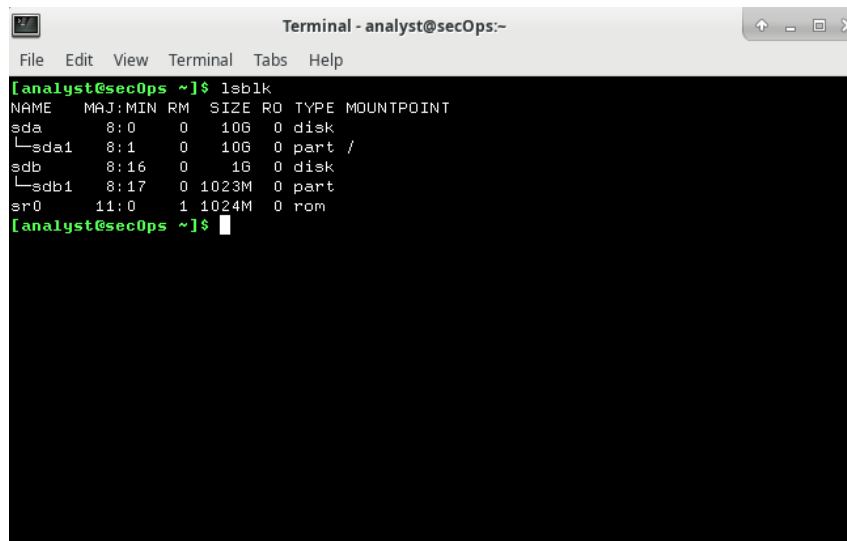
Navigating the Linux Filesystem and Permission Settings

Parte 1:

Esplorazione dei file system in Linux

Iniziamo il nostro laboratorio avviando la VM CyberOps Workstation e aprendo una finestra del terminale. I Filesystem devono esser “montati” prima di poter essere resi accessibili; montare un filesystem significa renderlo accessibile al sistema operativo.

Utilizzare il *lsblk* per visualizzare tutti i dispositivi a blocchi:



```
Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

[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
```

La VM CyberOps Workstation prevede installati tre dispositivi a blocchi: sr0, sda e sdb. L'output ad albero mostra anche le partizioni sotto sda e sdb.

Utilizziamo *mount* per visualizzare informazioni più dettagliate sui file system attualmente montati.

```
[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=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/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
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/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=36,prgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
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)
```

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

Eseguire *mount* nuovamente il comando, ma questa volta utilizzare il pipe per inviare l'output di mount a **grep** per filtrare l'output e visualizzare solo il file system radice:

```
[analyst@sec0ps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime,data=ordered)
```

Mount ci mostra che il filesystem root si trova nella prima partizione del dispositivo a blocchi sda (`/dev/sda1`). Sappiamo che questo è il filesystem root a causa del punto di montaggio utilizzato: `"/"`. L'output ci dice anche il tipo di formattazione utilizzato: `ext4` in questo caso. Le informazioni tra parentesi si riferiscono alle opzioni di montaggio della partizione.

Procediamo poi con i comandi *cd* e *ls -l*:

```
[analyst@sec0ps ~]$ cd /
[analyst@sec0ps /]$ 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 19 root root 3120 Oct 28 04:33 dev
drwxr-xr-x 58 root root 4096 Apr 17  2018 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  2 root root 4096 Jan  5  2018 opt
dr-xr-xr-x 117 root root    0 Oct 28 04:33 proc
drwxr-xr-x  7 root root 4096 Oct 23 06:42 root
drwxr-xr-x 17 root root  480 Oct 28 04:33 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 Oct 28 04:33 sys
drwxrwxrwt  8 root root  200 Oct 28 04:34 tmp
drwxr-xr-x  9 root root 4096 Apr 17  2018 usr
drwxr-xr-x 12 root root 4096 Apr 17  2018 var
```

Il primo comando cambia la directory nella directory root. La directory root è il livello più alto dei filesystem. Poiché `/dev/sda1` è montato sulla directory root (`"/"`), elencando i file nella directory root, l'utente sta in realtà elencando i file fisicamente archiviati nel root del filesystem `/dev/sda1`.

Montaggio e smontaggio manuale dei file system

Mount può anche essere utilizzato per montare e smontare i filesystem. Abbiamo visto che la VM CyberOps Workstation ha due dischi rigidi installati: il primo è stato riconosciuto dal kernel come `/dev/sda` mentre il secondo è stato riconosciuto come `/dev/sdb`. Prima che un dispositivo a blocchi possa essere montato, deve avere un punto di montaggio.

Utilizziamo *ls -l* per verificare che la directory *second_drive* si trovi nella directory home dell'analyst.

```
[analyst@secOps ~]$ cd ~
[analyst@secOps ~]$ ls -l
total 708
-rw-r--r-- 1 root    root      5563 Oct 23 05:06 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    581279 Oct 25 04:57 httpdump.pcap
-rw-r--r-- 1 root    root   118206 Oct 25 05:16 httpsdump.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
```

Utilizziamo *mount* per montare */dev/sdb1* sulla *second_drive* directory appena creata. La sintassi è: **mount** **[opzioni]** **<dispositivo da montare>** **<punto di montaggio>**.

Ora che */dev/sdb1* è stato montato su */home/analyst/second_drive*, utilizzare *ls -l* per elencare nuovamente il contenuto della directory.

```
[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
```

Smontare i filesystem è altrettanto semplice. Assicuriamoci di cambiare la directory in qualcosa al di fuori del punto di montaggio e usiamo *umount* come mostrato di seguito:

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

Parte 2:

Permessi dei file

I filesystem Linux hanno funzionalità integrate per controllare la capacità degli utenti di visualizzare, modificare, navigare ed eseguire i contenuti del filesystem. In sostanza, ogni file nei filesystem ha il suo set di permessi.

Considerate il file *cyops.mn* come esempio:

```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[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--*

Il proprietario del file (l'utente 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-). A tutti gli altri utenti non è consentito scrivere o eseguire quel file.

Touch è molto semplice e utile: permette la creazione rapida di un file di testo vuoto. Usiamolo per creare un file vuoto nella directory **/mnt**.

Il nuovo file è **myFile.txt**:

```
[analyst@sec0ps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
[analyst@sec0ps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
[analyst@sec0ps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@sec0ps scripts]$ cd ~/second_drive
[analyst@sec0ps 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@sec0ps second_drive]$ sudo chmod 665 myFile.txt
[analyst@sec0ps 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
```

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**.

Chmod viene utilizzato per modificare i permessi di un file o di una directory. Utilizziamo *Chmod* per modificare i permessi di **myFile.txt**.

Chown è usato per cambiare la proprietà di un file o di una directory. Emetti il comando seguente per rendere root il proprietario di **myFile.txt** :

```
[analyst@sec0ps second_drive]$ sudo chown analyst myFile.txt
[analyst@sec0ps 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
```

Per modificare contemporaneamente sia il proprietario che il gruppo in analyst , utilizzeremo il formato **sudo chown analyst:analyst myFile.txt** .

Directory e autorizzazioni

Anche le directory hanno permessi: sia i file che le directory hanno 9 bit per i permessi del proprietario/utente, del gruppo e di altri. Ci sono anche altri tre bit per permessi speciali: setuid, setgid e sticky.

Torniamo ora alla directory **/home/analyst/lab.support.files** ed esegui il **ls -l** per elencare tutti i file con i dettagli:

```
Terminal - analyst@secOps:~/lab.support.files
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ 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-tutorial1.log
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 malware
-rw-r--r-- 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 3 analyst analyst 4096 Mar 21 2018 scripts
-rw-r--r-- 1 analyst analyst 25553 Mar 21 2018 SQL_Lab.pcap
[analyst@secOps lab.support.files]$
```

Confrontiamo i permessi della directory **malware** con il file **mininet_services** e vediamo che prima delle autorizzazione per la directory del malware c'è la lettera *d*. Questa lettera all'inizio della riga indica che il tipo di file è una directory e non un file. 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. Una directory con il bit di esecuzione impostato ci segnala se un utente ha accesso o meno a quella directory.

I comandi *chmod*, *chown* funzionano per le directory nello stesso modo in cui funzionano per i file.

Collegamenti simbolici e altri tipi di file speciali

Il primo carattere in ogni file elencato con il comando *ls -l* mostra il tipo di file.

I tre diversi tipi di file in Linux, inclusi i loro sottotipi e caratteri, sono:

File regolari (-):

- File leggibili – file di testo
- File binari – programmi
- File immagine
- File compressi

File di directory (d):

- Cartelle

File speciali:

- **File di blocco (b)** – File utilizzati per accedere all'hardware fisico, come i punti di montaggio per accedere ai dischi rigidi.
- **File di dispositivo a caratteri (c)** – File che forniscono un flusso seriale di input e output.
- **Pipe files (p)** – Un file utilizzato per passare informazioni in cui i primi byte in entrata sono i primi byte in uscita. Questo è anche noto come FIFO (first in first out).
- **File di collegamento simbolico (l)** – File utilizzati per collegarsi ad altri file o directory. Esistono due tipi: collegamenti simbolici e collegamenti fisici.
- **File oSocket** : vengono utilizzati per passare informazioni da un'applicazione all'altra per comunicare tramite una rete.

```
[analyst@secOps ~]$ ls -l /dev/
total 0
crw-r--r-- 1 root root      10, 235 Oct 28 04:33 autofs
drwxr-xr-x 2 root root      140 Oct 28 04:33 block
drwxr-xr-x 2 root root      100 Oct 28 04:33 bsg
crw----- 1 root root      10, 234 Oct 28 04:33 btrfs-control
drwxr-xr-x 3 root root       60 Oct 28 04:33 bus
lrwxrwxrwx 1 root root        3 Oct 28 04:33 cdrom -> sr0
drwxr-xr-x 2 root root    2800 Oct 28 04:33 char
crw----- 1 root root        5,  1 Oct 28 04:33 console
lrwxrwxrwx 1 root root      11 Oct 28 04:33 core -> /proc/kcore
crw----- 1 root root      10,  61 Oct 28 04:33 cpu_dma_latency
crw----- 1 root root      10, 203 Oct 28 04:33 cuse
drwxr-xr-x 6 root root      120 Oct 28 04:33 disk
drwxr-xr-x 3 root root       80 Oct 28 04:33 dri
```

Creiamo ora dei link simbolici.

I link simbolici in Linux sono come le scorciatoie in Windows. Ci sono due tipi di link in Linux: link simbolici e hard link. La differenza è che un file di link simbolico punta al nome file di un altro file e un file di hard link punta al contenuto di un altro file. Creiamo ora due file usando *echo*:

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

Utilizziamo poi *ln -s* per creare un collegamento simbolico a **file1.txt** e *ln* per creare un collegamento fisico a **file2.txt**:

```
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
[analyst@secOps ~]$ ls -l
total 1060
-rw-r--r-- 1 root root      5563 Oct 23 05:06 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 Oct 29 03:54 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst    9 Oct 29 03:52 file1.txt
-rw-r--r-- 2 analyst analyst    5 Oct 29 03:52 file2hard
-rw-r--r-- 2 analyst analyst    5 Oct 29 03:52 file2.txt
-rw-r--r-- 1 root root    581279 Oct 25 04:57 httpdump.pcap
-rw-r--r-- 1 root root   118206 Oct 25 05:16 httpsdump.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
-rw-r--r-- 1 analyst analyst 345088 Oct 28 05:51 W32.Nimda.Amm.exe
[analyst@secOps ~]$
```

Notiamo come il file **file1symbolic** sia un collegamento simbolico con una / all'inizio della riga e un puntatore -> a **file1.txt** . Il **file2hard** punta, invece, agli stessi attributi e alla stessa posizione del blocco del disco come **file2.txt**.

Cambiamo ora nomi dei file originali: **file1.txt** e **file2.txt** e osserviamo cosa succede sui file collegati:

```
[analyst@sec0ps ~]$ mv file2.txt file2new.txt
[analyst@sec0ps ~]$ mv file1.txt file1new.txt
[analyst@sec0ps ~]$
```

```
[analyst@sec0ps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@sec0ps ~]$ cat file2hard
hard
```

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** .