

Traccia BW III

Lab- Navigating the Linux Filesystem and Permission Settings

Inizialmente, dopo aver creato un altro IDE Drive in UTM con dimensione di 1 GB, ho usato il comando `lsblk` per visualizzare tutti i dispositivi a blocchi: c'erano 2 blocchi `sda` e `sdb`, `sda` con una partizione `sda1`. Poi ho creato una partizione anche per `sdb`:

```
[analyst@sec0ps ~]$ 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 
[analyst@sec0ps ~]$ sudo fdisk /dev/sdb
[sudo] password for analyst:

Welcome to fdisk (util-linux 2.32).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x6643b349.

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[analyst@sec0ps ~]$ sudo partprobe /dev/sdb
sudo: partprobe: command not found
[analyst@sec0ps ~]$ sudo partx -a /dev/sdb
partx: /dev/sdb: error adding partition 1
[analyst@sec0ps ~]$ 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 
[analyst@sec0ps ~]$
```

Il filesystem root memorizzato in `/dev/sda1` è dove è memorizzato il sistema operativo Linux stesso. Di default tutti gli strumenti e i programmi sono memorizzati lì:

```
[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=2016060k,nr_inodes=504015,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
```

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

Come mostrato sopra, la radice si trova nella partizione di `sda`: `sda1`.

Quando andiamo nella directory root nel terminale e cerchiamo tutti i file, non vediamo dev/sdb1 lì. È perché non è attualmente montato:

```
[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 20 root root 3240 Feb 3 06:09 dev
drwxr-xr-x 58 root root 4096 Jan 31 07:37 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 115 root root 0 Feb 3 06:06 proc
drwxr-xr-x 7 root root 4096 Jan 29 10:17 root
drwxr-xr-x 18 root root 520 Feb 3 06:07 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 Feb 3 06:06 sys
drwxrwxrwt 8 root root 200 Feb 3 06:07 tmp
drwxr-xr-x 9 root root 4096 Apr 17 2018 usr
drwxr-xr-x 12 root root 4096 Apr 17 2018 var
[analyst@sec0ps /]$ cd ~
[analyst@sec0ps ~]$ ls -l
total 16
drwxr-xr-x 2 analyst analyst 4096 Jan 31 07:57 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
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
```

Quindi, per montare manualmente il filesystem, possiamo usare il seguente comando `sudo mount /dev/sdb1 ~/second_drive`. Nel mio caso ho avuto un errore, quindi ho usato il comando `sudo mkfs.ext4 /dev/sdb1` per risolverlo, che crea un nuovo filesystem Ext4 sulla partizione specificata e punta alla partizione in cui verrà creato il filesystem. Dopo di che, `/dev/sdb1` viene montato in `second_drive`:

```
[analyst@sec0ps ~]$ sudo mount /dev/sdb1 ~/second_drive/
mount: /home/analyst/second_drive: wrong fs type, bad option, bad superblock on /dev/sdb1, missing codepage or helper program, or other error.
[analyst@sec0ps ~]$ ls -l second_drive/
total 0
[analyst@sec0ps ~]$ sudo mount /dev/sdb1 second_drive/
mount: /home/analyst/second_drive: wrong fs type, bad option, bad superblock on /dev/sdb1, missing codepage or helper program, or other error.
[analyst@sec0ps ~]$ sudo mkfs.ext4 /dev/sdb1
mkfs2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 261888 4k blocks and 65536 inodes
Filesystem UUID: c5e77f03-6900-4c26-b942-492a1306bc2a
Superblock backups stored on blocks:
32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

[analyst@sec0ps ~]$ sudo mkdir -p /mnt/newdrive
[analyst@sec0ps ~]$ sudo mount /dev/sdb1 /mnt/newdrive
[analyst@sec0ps ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              2.0G  0  2.0G   0% /dev
run              2.0G  580K  2.0G   1% /run
/dev/sda1        9.8G  5.0G  4.4G  54% /
tmpfs            2.0G  0  2.0G   0% /dev/shm
tmpfs            2.0G  0  2.0G   0% /sys/fs/cgroup
tmpfs            2.0G  12K  2.0G   1% /tmp
tmpfs            395M  4.0K  395M   1% /run/user/1000
/dev/sdb1        991M  2.6M  922M   1% /mnt/newdrive
[analyst@sec0ps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[analyst@sec0ps ~]$ ls -l second_drive/
total 16
drwx----- 2 root root 16384 Feb 3 06:18 lost+found
```

Ho usato di nuovo il comando `mount` per vedere informazioni dettagliate sulla partizione `/dev/sdb1`. Poi ho smontato la partizione:

```
[analyst@sec0ps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /mnt/newdrive type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime,data=ordered)
[analyst@sec0ps ~]$ sudo umount /dev/sdb1
[sudo] password for analyst:
[analyst@sec0ps ~]$ ls -l second_drive/
total 0
[analyst@sec0ps ~]$
```

I filesystem Linux hanno funzionalità integrate per controllare la capacità degli utenti di visualizzare, modificare, navigare ed eseguire i contenuti del filesystem. Possiamo visualizzare e modificare i permessi. Ad esempio, se scriviamo `ls -l lab.support.files/scripts/`, possiamo vedere i permessi. Ad esempio, i permessi del proprietario e del gruppo di tutti i file e la directory in cui si trova `analyst`. A differenza di altri, il proprietario del file `cyops.mn` ha comandi di lettura e scrittura, ma non di esecuzione (`-rw-r--r--`). Per controllare i permessi della directory padre, ho provato a creare un file `test1.txt` usando il comando `touch`, tuttavia il permesso è stato negato. Questo perché la directory `mnt` ha i permessi per l'utente `root`. Quindi posso usare il comando `sudo` per creare il file sopra menzionato. Dopo aver creato il file, ho cambiato i permessi (`chmod 665`) e il proprietario (`chown analyst`) del file e ho aggiunto un testo al suo interno. Il `665` significa => il primo 6 è $4 + 2$, il secondo 6 è $4 + 2$, 5 è $4 + 1$. Quindi, come si vede, il numero dopo `chmod` è di tre cifre. Il primo è per i Proprietari, il secondo è per il Gruppo e l'ultimo è per gli Altri. 4 significa Lettura (r), 2 significa Scrittura (w) e 1 significa Esecuzione (x). Dare permessi completi significa `chmod 777` ($4+2+1=7$ per ciascuno):

```
[analyst@sec0ps 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
[analyst@sec0ps scripts]$ touch /mnt/test1.txt
touch: cannot touch '/mnt/test1.txt': Permission denied
[analyst@sec0ps scripts]$ ls -ld /mnt/
drwxr-xr-x 3 root root 4096 Feb  3 06:19 /mnt/
[analyst@sec0ps scripts]$ sudo touch /mnt/test1.txt
[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 16
drwx----- 2 root root 16384 Feb  3 06:18 lost+found
[analyst@sec0ps second_drive]$ sudo touch test2.txt
[analyst@sec0ps second_drive]$ ls -l
total 16
drwx----- 2 root root 16384 Feb  3 06:18 lost+found
-rw-r--r-- 1 root root 0 Feb  3 06:48 test2.txt
[analyst@sec0ps second_drive]$ sudo chmod 665 test2.txt
[sudo] password for analyst:
[analyst@sec0ps second_drive]$ sudo chown analyst test2.txt
[analyst@sec0ps second_drive]$ ls -l
total 16
drwx----- 2 root root 16384 Feb  3 06:18 lost+found
-rw-rw-r-x 1 analyst root 0 Feb  3 06:48 test2.txt
[analyst@sec0ps second_drive]$ echo test >> test2.txt
[analyst@sec0ps second_drive]$ cat test2.txt
test
```

Finora abbiamo esplorato i diversi tipi di file in Linux e come sono rappresentati nell'output del comando `ls -l`. Il primo carattere nell'elenco indica il tipo di file. Ma in generale, ci sono tre categorie principali di file:

- File regolari (-)
- Questi includono file di testo, che sono leggibili dall'uomo; file binari, come programmi eseguibili; file di immagine; e file compressi utilizzati per l'ottimizzazione dell'archiviazione.
- File di directory (d)
- Questi rappresentano cartelle che organizzano file e altre directory.
- File speciali
- I file di blocco (b) sono utilizzati per interagire con l'hardware, come i dispositivi di archiviazione montati.
- I file di dispositivo a caratteri (c) gestiscono input e output seriali, come i terminali tty.
- I file di pipe (p) consentono ai dati di fluire in modalità FIFO (first-in, first-out) tra i processi.
- I file di collegamento simbolico (l) creano riferimenti ad altri file o directory, funzionando come scorciatoie.
- I file socket (s) facilitano la comunicazione tra applicazioni su una rete.

Li osserviamo anche nella seguente foto:

```
[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_grandna.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
-rwxr-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 pcap
drwxr-xr-x 7 analyst analyst 4096 Mar 21 2018 pax
-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]$ ls -l /dev/
total 0
crw-r--r-- 1 root root 10, 235 Feb 3 06:07 autofs
drwxr-xr-x 2 root root 120 Feb 3 06:09 block
drwxr-xr-x 2 root root 80 Feb 3 06:06 bsg
crw----- 1 root root 10, 234 Feb 3 06:07 btrfs-control
drwxr-xr-x 3 root root 60 Feb 3 06:06 bus
drwxr-xr-x 2 root root 2980 Feb 3 06:07 char
crw----- 1 root root 5, 1 Feb 3 06:07 console
lrwxrwxrwx 1 root root 11 Feb 3 06:06 core -> /proc/kcore
crw----- 1 root root 10, 61 Feb 3 06:07 cpu_dna_latency
crw----- 1 root root 10, 203 Feb 3 06:07 cuse
drwxr-xr-x 6 root root 120 Feb 3 06:06 disk
drwxr-xr-x 3 root root 100 Feb 3 06:07 dri
crw-rw---- 1 root video 29, 0 Feb 3 06:07 fb0
lrwxrwxrwx 1 root root 13 Feb 3 06:06 fd -> /proc/self/fd
crw-rw-rw- 1 root root 1, 7 Feb 3 06:07 full
crw-rw-rw- 1 root root 10, 229 Feb 3 06:07 fuse
crw----- 1 root root 245, 0 Feb 3 06:07 hidraw0
crw----- 1 root root 245, 1 Feb 3 06:07 hidraw1
crw----- 1 root root 245, 2 Feb 3 06:07 hidraw2
crw-rw---- 1 root audio 10, 228 Feb 3 06:07 hpet
drwxr-xr-x 2 root root 0 Feb 3 06:06 hugepages
```

In Linux, i link simbolici funzionano come le scorciatoie in Windows. Esistono due tipi di link:

- Link simbolici (soft link) → Puntano al nome file di un altro file.
- Hard link → Puntano direttamente al contenuto di un altro file.

La differenza fondamentale è che se il file originale viene eliminato, un link simbolico si interrompe, mentre un hard link rimane intatto poiché fa riferimento agli stessi dati sul disco. Possiamo creare file usando echo e sperimentare con questi tipi di link. Ho usato `ln -s` per creare un link simbolico a `file1.txt` e `ln` per creare un hard link a `file2.txt`:

```
[analyst@secOps ~]$ echo "simbolico" > file1.txt
[analyst@secOps ~]$ echo "duro" > file2.txt
[analyst@secOps ~]$ cat file1.txt
simbolico
[analyst@secOps ~]$ cat file2.txt
duro
[analyst@secOps ~]$ ln -s file1.txt file1simbolico
[analyst@secOps ~]$ ln file2.txt file2duro
[analyst@secOps ~]$ ls -l
total 28
drwxr-xr-x 2 analyst analyst 4096 Jan 31 07:57 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
lrwxrwxrwx 1 analyst analyst 9 Feb 3 07:09 file1simbolico -> file1.txt
-rw-r--r-- 1 analyst analyst 10 Feb 3 07:07 file1.txt
-rw-r--r-- 2 analyst analyst 5 Feb 3 07:08 file2duro
-rw-r--r-- 2 analyst analyst 5 Feb 3 07:08 file2.txt
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
drwxr-xr-x 3 root root 4096 Feb 3 06:48 second_drive
[analyst@secOps ~]$ mv file1
file1simbolico file1.txt
[analyst@secOps ~]$ mv file1
file1simbolico file1.txt
[analyst@secOps ~]$ mv file1.txt
```

Ora cambiamo i nomi dei file originali: `file1.txt` e `file2.txt`, e osserviamo come influisce sui file collegati. Per questo ho usato il comando `mv`:

```
[analyst@secOps ~]$ mv file1.txt nuovofile1.txt
[analyst@secOps ~]$ mv file2.txt nuovofile2.txt
[analyst@secOps ~]$ cat file1simbolico
cat: file1simbolico: No such file or directory
[analyst@secOps ~]$ cat file2duro
duro
[analyst@secOps ~]$ ls -l
total 28
drwxr-xr-x 2 analyst analyst 4096 Jan 31 07:57 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
lrwxrwxrwx 1 analyst analyst 9 Feb 3 07:09 file1simbolico -> file1.txt
-rw-r--r-- 2 analyst analyst 5 Feb 3 07:08 file2duro
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
-rw-r--r-- 1 analyst analyst 10 Feb 3 07:07 nuovofile1.txt
-rw-r--r-- 2 analyst analyst 5 Feb 3 07:08 nuovofile2.txt
drwxr-xr-x 3 root root 4096 Feb 3 06:48 second_drive
[analyst@secOps ~]$
```

Abbiamo osservato come si comportano i collegamenti simbolici e fissi quando cambia il nome del file originale. Il collegamento simbolico `file1simbolico` si è interrotto perché puntava al nome file `file1.txt`, che è stato rinominato. Tuttavia, il collegamento fisso `file2duro` ha continuato a funzionare correttamente perché si collega direttamente all'inode di `file2.txt`, che è stato rinominato in `nuovofile2.txt`. Ciò ha confermato che i collegamenti simbolici dipendono dai nomi file, mentre i collegamenti fissi fanno riferimento alla struttura dati effettiva, rendendoli più resilienti alla ridenominazione.