Report: Navigazione nel Filesystem Linux e Impostazioni delle Permessioni

Report: Navigazione nel Filesystem Linux e Impostazioni delle Permessioni

Obiettivi
Questo laboratorio ha avuto l'obiettivo di acquisire familiarità con i filesystem Linux, comprendere le permessioni dei file e i collegamenti simbolici e altri tipi speciali di file.

Parte 1: Esplorazione del Filesystem Linux

Step 1: Accesso alla linea di comando

Ho avviato la CyberOps Workstation VM e aperto una finestra terminale.

Step 2: Visualizzazione dei filesystem montati

- Ho eseguito il comando `lsblk` per verificare i dispositivi a blocchi montati, ottenendo come output:

```
[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 ~]$
```

- Ho utilizzato `mount` per verificare i dettagli del filesystem montato.

```
[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/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=27,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10389)
mqueue on /dev/mqueue type mqueue (rw,relatime)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
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)
```

- Ho filtrato l'output con `mount | grep sda1`,
  confermando che il filesystem root è formattato in `ext4` e montato su `/`.

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

Step 3: Montaggio e smontaggio manuale dei filesystem

- Ho verificato la presenza della directory `second_drive` nella home dell'utente

```
[analyst@secOps ~]$ ls -l
total 17184
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     3489384 Jan 31 05:36 httpdump.pcap
-rw-r--r-- 1 root    root    14089118 Jan 31 06:11 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
```

- Ho montato `/dev/sdb1` su `~/second_drive` con:
  `sudo mount /dev/sdb1 ~/second_drive/`
- Ho verificato i contenuti della directory `second_drive`, confermando la presenza di file con:
  `ls -l second_drive/`

```
[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
```

- Ho dato di nuovo il comando `mount` filtrando l'output con `grep | /dev/sd`

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

- Ho smontato il dispositivo con:
  `sudo umount /dev/sdb1`

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

Parte 2: Permessi dei File

Step 1: Visualizzazione e Modifica dei Permessi

- Ho navigato nella directory `lab.support.files/scripts/` ed eseguito `ls -l` per visualizzare i permessi dei file.
- Ho analizzato i permessi del file `cyops.mn`: `-rw-r--r--`, confermando che solo l'utente proprietario può modificarlo.

```
[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
```

- Ho provato a creare un file in `/mnt` con `touch`, ottenendo un errore di permessi insufficienti.

```
[analyst@secOps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
```

- Ho verificato i permessi di `/mnt` con `ls -ld /mnt`, confermando che solo `root` può scrivere.

```
[analyst@secOps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
```

Come prima, ho montato la partizione `/dev/sdb1` sulla directory `/second_drive` creata in precedenza in questo laboratorio, ho elencato il contenuto della directory `second_drive` e ho visto i permessi del file `myFile.txt`

```
[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
```

- Ho modificato i permessi del file specifico `myFile.txt` con:

  `sudo chmod 665 myFile.txt`

  confermando che ora il proprietario e il gruppo possono leggerlo e modificarlo, mentre gli altri utenti possono solo leggerlo.

```
[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
```

Il `chmod` comando accetta i permessi in formato ottale. In questo modo, una ripartizione del 665 è la seguente:

6 in ottale è 110 in binario. Supponendo che ogni posizione dei permessi di un file possa essere 1 o 0, 110 significa rw- (lettura=1, scrittura=1 ed esecuzione=0).

Pertanto, il comando `chmod 665 myFile.txt` modifica i permessi in:

Proprietario: rw- (6 in ottale o 110 in binario)
Gruppo: rw- (6 in ottale o 110 in binario)
Altro: rx (5 in ottale o 101 in binario)

- Ho cambiato il proprietario del file con:
  `sudo chown analyst myFile.txt`
  permettendo all'utente `analyst` di modificarlo.
- Ho testato la modifica del file con il comando:
  `echo "test" >> myFile.txt`

```
[analyst@secOps second_drive]$ echo test >> 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
```

verificando che l'operazione fosse riuscita grazie ai permessi correttamente impostati.

Step 2: Permessi delle Directory

Similmente ai file normali, 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.
Sono tornato alla directory `/lab.support.files` e ho eseguito `ls -l` per elencare tutti i file con i dettagli:

```
[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
-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 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
```

- Ho confrontato i permessi di una directory (`malware`) e un file (`mininet_services`), osservando che le directory iniziano con `d`.
  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.

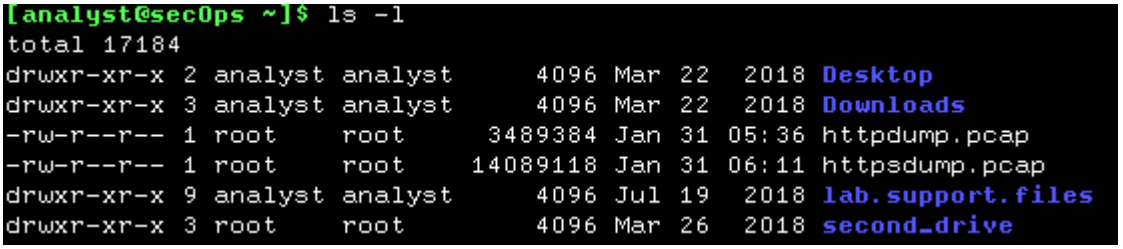
- I comandi `chmod` e `chown` per modificare rispettivamente i permessi e il proprietario della directory, funzionano sia per i file e sia per le directory allo stesso modo.

Questa parte del laboratorio ha permesso di comprendere in modo approfondito la gestione delle autorizzazioni nei file e nelle directory, evidenziando come un'impostazione errata possa influenzare l'accesso e la sicurezza del sistema.

Parte 3: Collegamenti Simbolici e Altri Tipi di File

Step 1: Esaminare i tipi di file

- Ho eseguito `ls -l` in `/analyst/` per verificare i file e le directory.



- Ho esaminato i file speciali in `/dev/` con `ls -l /dev/`,

```
[analyst@secOps ~]$ ls -l /dev/
total 0
crw-r--r--  1 root root       10, 235 Feb  3 04:22 autofs
drwxr-xr-x  2 root root            140 Feb  3 04:22 block
drwxr-xr-x  2 root root            100 Feb  3 04:22 bsg
crw-------  1 root root       10, 234 Feb  3 04:22 btrfs-control
drwxr-xr-x  3 root root             60 Feb  3 04:22 bus
lrwxrwxrwx  1 root root              3 Feb  3 04:22 cdrom -> sr0
drwxr-xr-x  2 root root           2800 Feb  3 04:23 char
crw-------  1 root root        5,    1 Feb  3 04:23 console
lrwxrwxrwx  1 root root             11 Feb  3 04:22 core -> /proc/kcore
crw-------  1 root root       10,   61 Feb  3 04:22 cpu_dma_latency
crw-------  1 root root       10,  203 Feb  3 04:22 cuse
drwxr-xr-x  6 root root            120 Feb  3 04:22 disk
drwxr-xr-x  3 root root             80 Feb  3 04:22 dri
crw-rw----  1 root video      29,    0 Feb  3 04:22 fb0
lrwxrwxrwx  1 root root             13 Feb  3 04:22 fd -> /proc/self/fd
crw-rw-rw-  1 root root        1,    7 Feb  3 04:22 full
crw-rw-rw-  1 root root       10,  229 Feb  3 04:22 fuse
crw-------  1 root root      245,    0 Feb  3 04:22 hidraw0
crw-rw----  1 root audio      10,  228 Feb  3 04:22 hpet
drwxr-xr-x  2 root root             0 Feb  3 04:22 hugepages
lrwxrwxrwx  1 root root             25 Feb  3 04:22 initctl -> /run/systemd/initctl/fifo
drwxr-xr-x  4 root root            360 Feb  3 04:22 input
crw-r--r--  1 root root        1,   11 Feb  3 04:22 kmsg
drwxr-xr-x  2 root root             60 Feb  3 04:22 lightnvm
lrwxrwxrwx  1 root root             28 Feb  3 04:22 log -> /run/systemd/journal/dev-log
crw-rw----  1 root disk       10, 237 Feb  3 04:22 loop-control
drwxr-xr-x  2 root root             60 Feb  3 04:22 mapper
crw-r-----  1 root kmem        1,    1 Feb  3 04:22 mem
crw-------  1 root root       10,   58 Feb  3 04:22 memory_bandwidth
drwxrwxrwt  2 root root             40 Feb  3 04:22 mqueue
drwxr-xr-x  2 root root             60 Feb  3 04:22 net
crw-------  1 root root       10,   60 Feb  3 04:22 network_latency
crw-------  1 root root       10,   59 Feb  3 04:22 network_throughput
crw-rw-rw-  1 root root        1,    3 Feb  3 04:22 null
crw-r-----  1 root kmem        1,    4 Feb  3 04:22 port
crw-------  1 root root      108,    0 Feb  3 04:22 ppp
crw-------  1 root root       10,    1 Feb  3 04:22 psaux
crw-rw-rw-  1 root tty         5,    2 Feb  3 06:01 ptmx
drwxr-xr-x  2 root root             0 Feb  3 04:22 pts
crw-rw-rw-  1 root root        1,    8 Feb  3 04:22 random
lrwxrwxrwx  1 root root             4 Feb  3 04:22 rtc -> rtc0
```

- osservando che:
  - `b` indica file a blocchi (dischi rigidi, USB, etc.).
  - `c` indica dispositivi a caratteri (terminali, dispositivi input/output).
  - `l` indica collegamenti simbolici.

Step 2: Creazione e Manipolazione dei Collegamenti

- Ho creato due file con il comando `echo`:

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

- Poi ho creato un collegamento simbolico al `file1.txt` e un collegamento fisico al `file2.txt` con questi comandi:
  `ln -s file1.txt file1simbolico`

`ln file2.txt file2difficile`

con `ls -l` ho elencato:

```
[analyst@secOps ~]$ ln -s file1.txt file1simbolico
[analyst@secOps ~]$ ln file2.txt file2difficile
[analyst@secOps ~]$ ls -l
total 17196
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 Feb  3 06:16 file1simbolico -> file1.txt
-rw-r--r-- 1 analyst analyst       10 Feb  3 06:06 file1.txt
-rw-r--r-- 2 analyst analyst       10 Feb  3 06:07 file2difficile
-rw-r--r-- 2 analyst analyst       10 Feb  3 06:07 file2.txt
-rw-r--r-- 1 root    root     3489384 Jan 31 05:36 httpdump.pcap
-rw-r--r-- 1 root    root    14089118 Jan 31 06:11 httpsdump.pcap
drwxr-xr-x 9 analyst analyst     4096 Jul 19  2018 lab.support.files
```

Ho notato come il file file1simbolico sia un collegamento simbolico con una `l` all'inizio della riga e un puntatore `->` a file1.txt . Il file2difficile sembra essere un file normale, perché in effetti è un file normale che punta allo stesso inode sul disco rigido come file2.txt . In altre parole, file2difficile punta agli stessi attributi e alla stessa posizione del blocco del disco come file2.txt . Il numero 2 nella quinta colonna dell'elenco per file2difficile e file2.txt indica che ci sono 2 file collegati in modo fisso allo stesso inode.

- Dopo aver rinominato `file1.txt` in `file1new.txt`, il collegamento simbolico `file1simbolico` è diventato non valido, mentre `file2difficile` ha continuato a funzionare.

```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
[analyst@secOps ~]$ cat file1simbolico
cat: file1simbolico: No such file or directory
[analyst@secOps ~]$ cat file2difficile
difficile
```

- Quindi ho notato che `file1simbolico` è ora un collegamento simbolico non funzionante perché il nome del file a cui puntava `file1.txt` è cambiato, mentre il file di collegamento fisso `file2difficile` funziona ancora correttamente perché punta all'inode di `file2.txt` e non al suo nome, che ora è `file2new.txt` .

Riflessione

Durante questo laboratorio, ho approfondito la gestione del filesystem Linux, imparando a montare e smontare dispositivi, a modificare i permessi dei file e a distinguere tra collegamenti simbolici e hard link.

La gestione corretta delle autorizzazioni e dei permessi è fondamentale per la sicurezza e il buon funzionamento di un sistema Linux. Un file con permessi troppo permissivi potrebbe esporre dati sensibili, mentre permessi troppo restrittivi potrebbero impedire l'esecuzione di programmi essenziali. Inoltre, comprendere la differenza tra collegamenti simbolici e hard link è cruciale per gestire riferimenti ai file senza duplicare inutilmente lo spazio disco.

Queste competenze sono essenziali per qualsiasi amministratore di sistema o professionista della sicurezza informatica, poiché garantiscono il controllo sull'accesso ai file e la protezione delle

informazioni, evitando errori comuni che possono compromettere l'integrità e la sicurezza di un sistema.