Report Esercizio 3: Navigazione del Filesystem Linux e Gestione dei Permessi

A cura di Iris Canole, Federico Giannini, Daniele Castello, Luca Pani, Rosario Papa, Yari Olmi, Alessandro Salerno

Introduzione

Questo report ha lo scopo di documentare e analizzare i passaggi eseguiti durante l'esercizio di laboratorio n. 3. L'attività si è focalizzata su aspetti fondamentali dell'ambiente Linux, tra cui l'esplorazione della struttura del filesystem, la gestione dei permessi di accesso a file e directory, e la comprensione delle diverse tipologie di file, con un'attenzione particolare ai link simbolici e hard. L'obiettivo è consolidare le competenze pratiche necessarie per operare in modo sicuro ed efficiente su un sistema operativo Linux.

---

1. Parte 1: Esplorazione dei Filesystem in Linux

Una comprensione approfondita della struttura e della gestione dei filesystem è fondamentale per qualsiasi operazione in ambiente Linux, dalla semplice navigazione all'amministrazione di sistema e all'analisi della sicurezza. Questa sezione documenta le procedure di identificazione, montaggio e analisi dei dispositivi di archiviazione disponibili nella macchina virtuale utilizzata per il laboratorio.

1.1 Analisi dei Dispositivi a Blocchi

Per visualizzare i dispositivi di archiviazione, noti come dispositivi a blocchi, è stato utilizzato il comando `lsblk`. Questo strumento offre una panoramica chiara e strutturata dell'hardware di memorizzazione collegato al sistema.
L'output del comando ha rivelato la presenza di tre dispositivi principali: `sda`, `sdb` e `sr0`. Nello specifico:

- **sda**: Un disco da 10GB, contenente una singola partizione (sda1).
- **sdb**: Un disco da 1GB, anch'esso con una singola partizione (sdb1).
- **sr0**: Un dispositivo di tipo ROM, probabilmente un'unità CD/DVD virtuale.

La partizione `sda1` risulta già montata sulla directory radice (`/`), indicando che ospita il filesystem principale del sistema operativo.

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

1.2 Analisi dei Filesystem Montati

Per ispezionare quali filesystem sono attualmente accessibili dal sistema operativo, è stato utilizzato il comando `mount`. Per focalizzare l'analisi sul filesystem principale, l'output è stato filtrato tramite `grep`, isolando le informazioni relative alla partizione `sda1`.

L'analisi dell'output ha confermato che:

- La partizione `/dev/sda1` è montata sul punto di montaggio `/` (la radice del filesystem).
- Il tipo di filesystem è `ext4`, uno standard comune per le distribuzioni Linux.
- Le opzioni di montaggio includono `rw` (read-write), che indica che il filesystem è accessibile sia in lettura che in scrittura, e relatime, un'opzione che ottimizza le performance aggiornando i tempi di accesso ai file in modo più efficiente rispetto al default.

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

1.3 Ispezione della Directory Home e Posizione Fisica dei File

È stato eseguito il comando `ls -l` all'interno della directory home dell'utente (`/home/analyst`) per esaminarne il contenuto e i relativi metadati.

```
[analyst@secOps ~]$ cd
[analyst@secOps ~]$ ls -l
total 32
-rw-r--r-- 1 root    root    5442 Dec  2 09:28 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Jun 17 19:35 Desktop
drwxr-xr-x 3 analyst analyst 4096 Jun 18 20:17 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jun 18 20:17 lab.support.files
drwxr-xr-x 3 analyst analyst 4096 Jun 18 19:55 scripts
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 second_drive
drwxr-xr-x 5 analyst analyst 4096 Jun 18 19:27 yay
```

Qual è il significato dell'output? L'output del comando `ls -l` fornisce informazioni dettagliate su ogni file e directory, strutturate in colonne:

1. **Tipo e Permessi**: Il primo carattere indica il tipo (`-` per file, `d` per directory), seguito da tre set

di permessi (lettura, scrittura, esecuzione) per il proprietario, il gruppo e gli altri utenti.
2. **Proprietario e Gruppo**: I nomi dell'utente e del gruppo che possiedono il file.
3. **Dimensione**: La dimensione in byte.
4. **Data di Modifica**: La data e l'ora dell'ultima modifica.
5. **Nome**: Il nome del file o della directory.

Dove sono fisicamente memorizzati i file elencati? I file sono fisicamente memorizzati sul dispositivo a blocchi `/dev/sda`. A livello logico, risiedono all'interno del filesystem `ext4` che è stato creato sulla partizione `/dev/sda1` e montato sulla directory radice (`/`).

Perché `/dev/sdb1` non viene mostrato nell'output sopra? Il dispositivo `/dev/sdb1` non appare nell'elenco perché `ls -l` mostra il contenuto della directory corrente, non l'elenco dei dispositivi hardware. In questo momento, `/dev/sdb1` è un dispositivo a blocchi riconosciuto dal sistema ma non ancora montato, quindi il suo contenuto non è accessibile attraverso il filesystem.

1.4 Montaggio e Smontaggio Manuale di un Filesystem

Questa fase ha dimostrato il processo di montaggio manuale della partizione `/dev/sdb1` su un punto di montaggio designato, la directory `second_drive`. Inizialmente, la directory `second_drive` era vuota.

```
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

Successivamente, la partizione è stata montata utilizzando il comando `sudo mount /dev/sdb1 ~/second_drive/`. Eseguendo nuovamente il comando `ls -l` sulla directory `second_drive`, è stato possibile osservare il contenuto del filesystem appena montato, che non era più vuoto.

```
[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
```

Perché la directory non è più vuota? La directory non è più vuota perché ora funge da punto di montaggio per il filesystem presente sulla partizione `/dev/sdb1`. Il contenuto visibile appartiene a quel filesystem, reso accessibile tramite la directory `second_drive`.

Dove sono fisicamente memorizzati i file elencati? I file `lost+found` e `myFile.txt` sono fisicamente memorizzati sul dispositivo hardware `/dev/sdb`.

Per verificare lo stato del sistema dopo il montaggio, sono stati eseguiti nuovamente i comandi `mount` e `lsblk`. L'output ha confermato che `/dev/sdb1` era correttamente montato su `/home/analyst/second_drive`.

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

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

Infine, il filesystem è stato smontato con il comando `sudo umount /dev/sdb1`. Una verifica successiva ha mostrato che la directory `second_drive` era tornata al suo stato originale vuoto, dimostrando che il collegamento tra la directory e il dispositivo era stato rimosso.

```
[analyst@secOps ~]$ sudo umount /dev/sdb1
[sudo] password for analyst:
```

```
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

La padronanza delle operazioni di montaggio e smontaggio è essenziale per gestire i dispositivi di archiviazione e costituisce la base per il controllo degli accessi ai dati, argomento centrale della prossima sezione.

---

2. Parte 2: Gestione dei Permessi dei File

Il modello di sicurezza di Linux si basa in modo critico sui permessi associati a file e directory. Una corretta configurazione di proprietà e permessi è essenziale non solo per proteggere i dati da accessi non autorizzati, ma anche per garantire il corretto funzionamento delle applicazioni che necessitano di leggere o scrivere su file specifici.

2.1 Analisi dei Permessi di un File

È stato analizzato il file `cyops.mn` all'interno della directory `lab.support.files/scripts/` per comprendere la struttura dei suoi permessi.

```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[analyst@secOps scripts]$ ls -l
total 68
-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 4053 Jun 18 20:09 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 5016 Jun 18 20:07 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 4189 Jun 18 19:22 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
```

Chi è il proprietario del file? E il gruppo? Dall'output di `ls -l`, si evince che sia il proprietario (`user`) sia il gruppo (`group`) del file `cyops.mn` sono `analyst`.

I permessi per `cyops.mn` **sono** `-rw-r--r--`**. Cosa significa?** La stringa dei permessi si scompone come segue:

- `-`: Indica che si tratta di un file regolare.
- `rw-`: Il proprietario (`analyst`) ha i permessi di lettura (`r`) e scrittura (`w`), ma non di esecuzione (`-`).
- `r--`: I membri del gruppo (`analyst`) hanno solo il permesso di lettura (`r`).
- `r--`: Tutti gli altri utenti del sistema hanno anch'essi solo il permesso di lettura (`r`).

2.2 Gestione dei Permessi di Scrittura su Directory

È stato tentato di creare un file vuoto nella directory `/mnt` tramite il comando `touch`. L'operazione è fallita, restituendo un errore "Permission denied".

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

Per investigare la causa dell'errore, sono stati ispezionati i permessi della directory `/mnt` con il comando `ls -ld /mnt`.

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

Perché il file non è stato creato? Il file non è stato creato perché la directory `/mnt` è di proprietà dell'utente `root` e del gruppo `root`, con permessi `drwxr-xr-x`. L'utente `analyst`, non essendo `root` (il proprietario) e non appartenendo al gruppo `root`, ricade nella categoria "altri utenti" (`others`). I permessi per questa categoria sono `r-x` (lettura ed esecuzione), ma manca il permesso di scrittura (`w`), che è indispensabile per creare, modificare o eliminare file all'interno di una directory.

Cosa si può fare affinché il comando touch mostrato sopra abbia successo? Esistono due soluzioni principali per completare l'operazione con successo:

1. **Usare** `sudo`: Eseguire il comando con privilegi di superutente (`sudo touch /mnt/myNewFile.txt`) per bypassare i limiti di permesso dell'utente analyst.
2. **Modificare i permessi**: Modificare i permessi della directory `/mnt` per concedere il permesso di scrittura all'utente analyst o al gruppo a cui appartiene.

2.3 Modifica dei Permessi con `chmod`

Dopo aver montato nuovamente la partizione `/dev/sdb1`, è stato esaminato il file `myFile.txt` al suo interno.

```
[analyst@secOps mnt]$  sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps mnt]$ 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
```

Quali sono i permessi del file myFile.txt? I permessi iniziali del file erano `-rw-r--r--` e il proprietario era l'utente `analyst`. Questo significa che il proprietario (`analyst`) poteva leggere e scrivere, mentre il gruppo (`analyst`) e gli altri utenti potevano solo leggere.

Successivamente, i permessi sono stati modificati utilizzando il comando `sudo chmod 665 myFile.txt`.

```
[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
```

I permessi sono cambiati? Quali sono i permessi di `myFile.txt`? Sì, i permessi sono stati aggiornati con successo. I nuovi permessi sono `-rw-rw-r-x`. Questo significa che:

- Il proprietario (`analyst`) può leggere e scrivere.
- I membri del gruppo (`analyst`) possono ora anche scrivere sul file.
- Gli altri utenti possono leggere ed eseguire il file.

Quale comando cambierebbe i permessi di `myFile.txt` a `rwxrwxrwx`? Per concedere pieno accesso (lettura, scrittura ed esecuzione) a tutti gli utenti, il comando da utilizzare sarebbe `chmod 777 myFile.txt`.

2.4 Modifica della Proprietà con `chown`

È stato utilizzato il comando `chown` per cambiare il proprietario e il gruppo del file `myFile.txt` a `analyst`.

```
[analyst@secOps second_drive]$  sudo chown analyst: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
```

Una volta eseguito il comando, si è tentato di aggiungere del testo al file, verificandone poi il contenuto.

```
[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
```

L'operazione è riuscita? Spiega. Sì, l'operazione di scrittura è riuscita. L'analisi rivela un punto interessante: l'utente `analyst` era *già* il proprietario del file prima dell'esecuzione del comando `chown`. Pertanto, il comando `sudo chown analyst:analyst myFile.txt` è stato funzionalmente ridondante, in quanto non ha modificato la proprietà esistente. La scrittura ha avuto successo perché l'utente `analyst`, in qualità di proprietario, possedeva già il permesso di scrittura (w) fin dall'inizio, come indicato dai permessi originali `-rw-r--r--`.

2.5 Differenze tra Permessi di File e Directory

Analizzando l'output di `ls -l` sulla directory `lab.support.files`, è possibile osservare le differenze tra file regolari e directory.

```
[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 6 analyst analyst   4096 Aug 15  2022 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 Jun 18 20:07 scripts
-rw-r--r-- 1 analyst analyst  25553 Mar 21  2018 SQL_Lab.pcap
```

Qual è la differenza tra la parte iniziale della riga di malware e la riga di `mininet_services`? La differenza fondamentale risiede nel primo carattere della stringa dei permessi:

- **malware**: La riga inizia con una d, che la identifica come una **directory**.
- **mininet_services**: La riga inizia con un `-`, che lo identifica come un **file regolare**.

Un'altra distinzione importante riguarda il bit di esecuzione (`x`). Per un file, questo bit permette che venga eseguito come programma. Per una directory, il bit di esecuzione conferisce il permesso di "entrare" o attraversare quella directory per accedere al suo contenuto.

Oltre ai permessi, Linux classifica i file in diverse tipologie con funzioni specializzate, che verranno esplorate nella prossima sezione.

---

3. Parte 3: Link Simbolici e Altri Tipi di File Speciali

Il filesystem Linux non è composto solo da file regolari e directory. Esistono infatti diverse tipologie di "file speciali", come i file di dispositivo e i link, che svolgono ruoli cruciali nell'interazione con l'hardware e nell'organizzazione efficiente dei dati.

3.1 Esame dei Tipi di File

Un'analisi della directory home dell'utente (`/home/analyst`) ha mostrato la presenza dei tipi di file più comuni: file regolari (`-`) e directory (`d`).

```
[analyst@secOps ~]$ ls -l /home/analyst
total 32
-rw-r--r-- 1 root    root    5442 Dec  2 09:28 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Jun 17 19:35 Desktop
drwxr-xr-x 3 analyst analyst 4096 Jun 18 20:17 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jun 18 20:17 lab.support.files
drwxr-xr-x 3 analyst analyst 4096 Jun 18 19:55 scripts
drwxr-xr-x 3 root    root    4096 Mar 26  2018 second_drive
drwxr-xr-x 5 analyst analyst 4096 Jun 18 19:27 yay
```

Un'ispezione della directory /dev, che gestisce l'interazione con i dispositivi hardware, ha invece rivelato la presenza di file speciali:

- **File a Blocco (`b`)**: Identificano dispositivi di archiviazione come `sdb1`.
- **File a Carattere (`c`)**: Gestiscono dispositivi che operano con flussi di dati seriali, come `snapshot`.
- **Link Simbolici (`l`)**: Puntatori ad altri file o directory, come stderr che punta a `/proc/self/fd/2`.

```
brw-rw----  1 root disk        8,  17 Dec  9 05:19 sdb1
drwxrwxrwt  2 root root           40 Dec  9 05:19 shm
crw-------  1 root root       10, 231 Dec  9 05:19 snapshot
drwxr-xr-x  3 root root          180 Dec  9 05:19 snd
brw-rw----+ 1 root optical    11,   0 Dec  9 05:19 sr0
lrwxrwxrwx  1 root root           15 Dec  9 05:18 stderr -> /proc/self/fd/2
```

3.2 Creazione e Confronto tra Link Simbolici e Hard Link

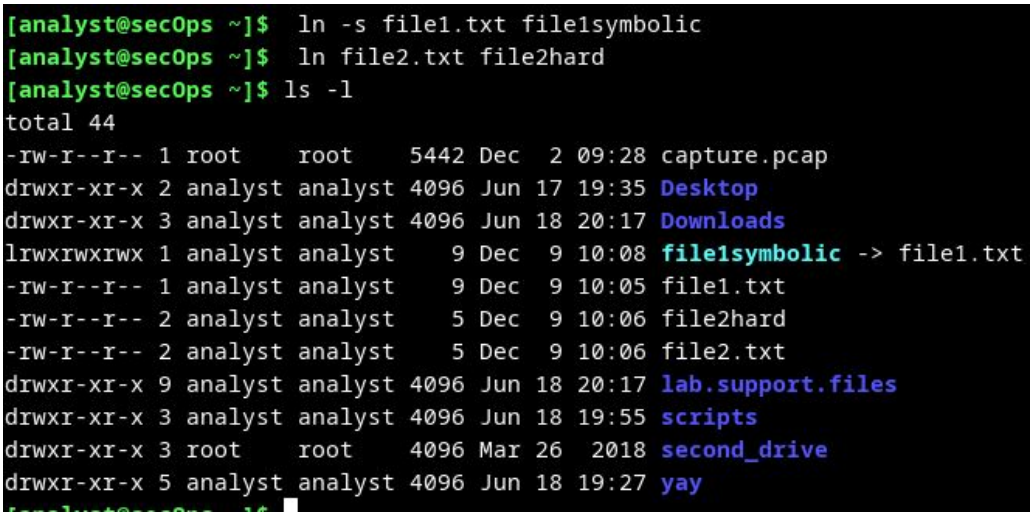
Per analizzare le differenze pratiche tra link simbolici e hard link, sono stati creati due file di origine: `file1.txt` con il contenuto "symbolic" e `file2.txt` con il contenuto "hard".

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

Successivamente, è stato creato un link simbolico (`file1symbolic`) puntato a `file1.txt` e un hard link (`file2hard`) collegato a `file2.txt`. L'output del comando `ls -l` ha evidenziato le seguenti differenze chiave:

- **Link Simbolico**: `file1symbolic` è identificato dal tipo di file l e mostra esplicitamente un puntatore (➜ `file1.txt`) al nome del file originale.

- **Hard Link**: `file2hard` appare come un file regolare. La differenza è visibile nel contatore di link (seconda colonna), che per `file2.txt` e `file2hard` è incrementato a 2, indicando che due nomi di file puntano allo stesso contenuto sul disco.



3.3 Comportamento dei Link dopo la Rinominazione dei File Originali

Per testare la resilienza dei due tipi di link, i file originali sono stati rinominati. Successivamente, si è tentato di leggere il contenuto di entrambi i link.

I risultati sono stati i seguenti:

- Il tentativo di leggere `file1symbolic` è fallito con l'errore "No such file or directory". Questo accade perché il link simbolico punta al *nome* del file originale. Avendo rinominato `file1.txt`, il puntatore è diventato "rotto".
- Il tentativo di leggere `file2hard` ha avuto successo, mostrando correttamente il contenuto "hard". Questo perché l'hard link punta direttamente all'*inode* (l'indice dei dati sul disco), non al nome del file. Di conseguenza, rimane valido anche se il nome originale del file cambia.

Cosa pensi succederebbe a file2hard se aprissi un editor di testo e cambiassi il testo in file2new.txt? Poiché `file2hard` e `file2new.txt` puntano allo stesso inode, e quindi alla stessa area dati sul disco, qualsiasi modifica apportata al contenuto di `file2new.txt` si rifletterebbe immediatamente anche leggendo il contenuto di `file2hard`. I due nomi sono semplicemente due

riferimenti diversi agli stessi identici dati.

Conclusioni

Questo laboratorio ha permesso di acquisire competenze pratiche essenziali per operare su sistemi Linux. La gestione dei permessi e della proprietà dei file è emersa come uno degli aspetti fondamentali della sicurezza e della stabilità del sistema, essendo una causa comune di problemi applicativi se non configurata correttamente. Infine, l'analisi dei diversi tipi di file, in particolare la distinzione funzionale tra link simbolici e hard link, ha consolidato la comprensione di come il filesystem Linux organizza e referenzia i dati in modo flessibile e potente.