REPORT TECNICO

Permessi di Linux

Redatto da: *Nicolò Calì Cybersecurity Student*

Data: *10/02/2026*

Oggetto: *Configurazione dei permessi di accesso per file sensibili su sistema Kali Linux.*

1. Introduzione

Il presente report documenta le attività svolte per la messa in sicurezza di file contenenti informazioni sensibili all'interno di un sistema operativo Linux.

L'esercitazione è stata condotta utilizzando una **Virtual Machine Kali Linux**, distribuzione standard per attività di penetration testing e security auditing.

L'obiettivo principale è quello di dimostrare la gestione pratica dei **permessi** (lettura, scrittura, esecuzione) e l'applicazione del principio del **Least Privilege** (Privilegio Minimo), garantendo che solo l'utente proprietario possa accedere ai dati riservati.

2. Creazione e Verifica Iniziale

Come primo passo, è stato creato un file di testo denominato **`top_secret.txt`** contenente dati simulati ad alta criticità. Poiché il file deve essere protetto, la creazione è avvenuta tramite privilegi amministrativi utilizzando il comando `sudo`.

Procedura:

- **Creazione** del file con l'editor di testo `nano` e privilegi di root.
- **Verifica** dei permessi predefiniti assegnati dal sistema tramite il comando **`ls -l`**.

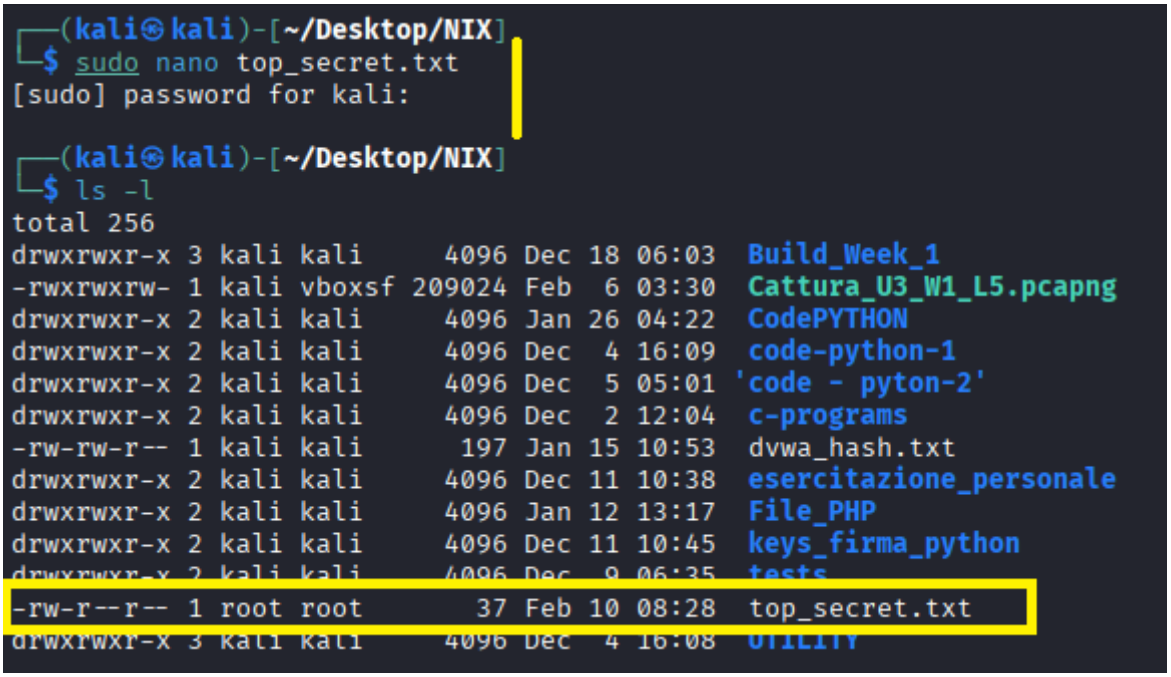

Fig. 1 Creazione del file .txt

Dall'output del comando **`ls -l`**, rileviamo la stringa dei permessi: **`-rw-r--r--`**

Questa configurazione indica che:

- **Proprietario (root)**: Ha permessi di lettura e scrittura (**`rw-`**).
- **Gruppo (root)**: Ha permessi di sola lettura (**`r--`**).
- **Altri (Others)**: Hanno permessi di sola lettura (**`r--`**).

Questa configurazione non è sicura per un file confidenziale.

I permessi di lettura assegnati alla categoria "Altri" permettono a qualsiasi utente del sistema di visualizzare il contenuto del file, esponendo le informazioni sensibili.

3. Modifica dei Permessi

Per mitigare il rischio identificato, si è proceduto all'**hardening** del file modificando i permessi di accesso. L'obiettivo è rimuovere qualsiasi diritto di lettura o scrittura per il gruppo e per gli altri utenti.

È stato utilizzato il comando **`sudo chmod 600 top_secret.txt`**.

Spiegazione della notazione 600:

- **6 (Proprietario):** Il proprietario mantiene il controllo totale.
- **0 (Gruppo):** Nessun permesso.
- **0 (Altri):** Nessun permesso.

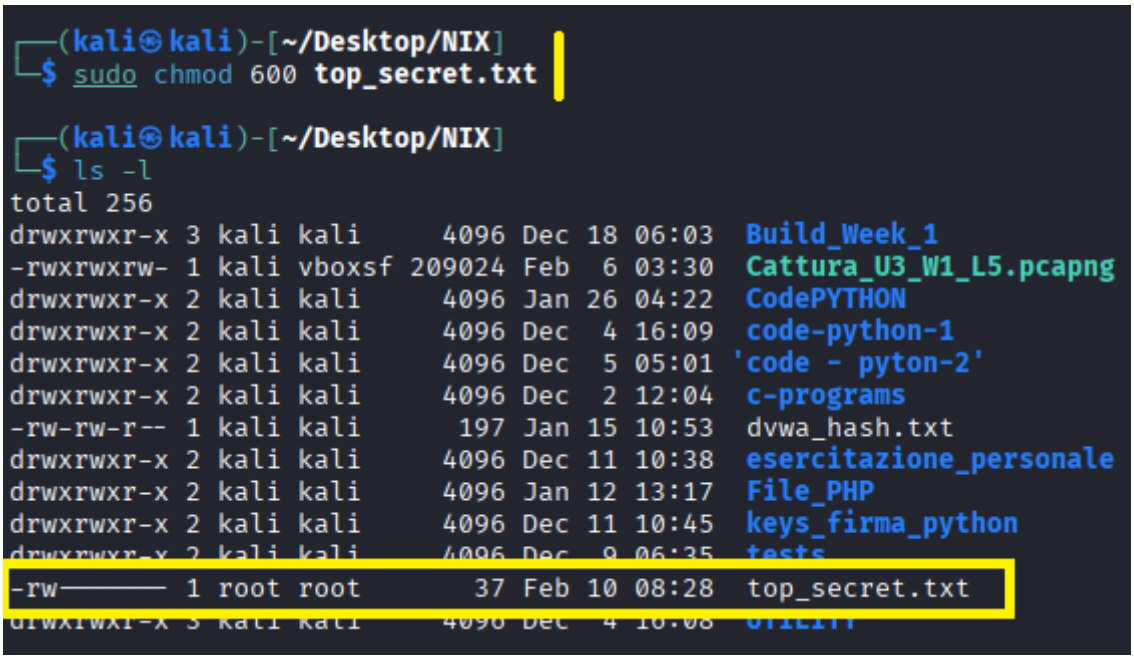

Fig. 2 Modifica permessi

Come mostrato in Figura 2, la nuova stringa dei permessi è: **`-rw-------`**

Ora il file è accessibile esclusivamente dall'utente **root**.

4. Test di Sicurezza

Per verificare l'efficacia delle modifiche applicate, è stato eseguito un test di accesso impersonando un utente non privilegiato.

È stato tentato di leggere il contenuto del file utilizzando il comando **`cat`**.

Fig. 3 Permission denied

Il sistema ha restituito l'errore: **`Permission denied`**.

Poiché l'utente `kali` non è il proprietario del file e rientra nella categoria "Altri" (i cui permessi sono ora impostati a **`---`**).

Il kernel di Linux ha correttamente bloccato la richiesta di lettura.

5. Extra: Scenario script pericoloso

Per completare l'analisi dei permessi, è stato simulato uno scenario in cui è necessario gestire un file eseguibile (script).

È stato creato un file denominato **`malware_test.sh`** contenente un semplice comando bash per simulare un potenziale script malevolo o un programma di sistema.

Procedura:

- Creazione del file script.
- Applicazione dei permessi restrittivi **`600`** (`rw-------`) come fatto nell'esercizio precedente.
- Tentativo di esecuzione dello script invocandolo direttamente da terminale con **`./malware_test.sh`**.

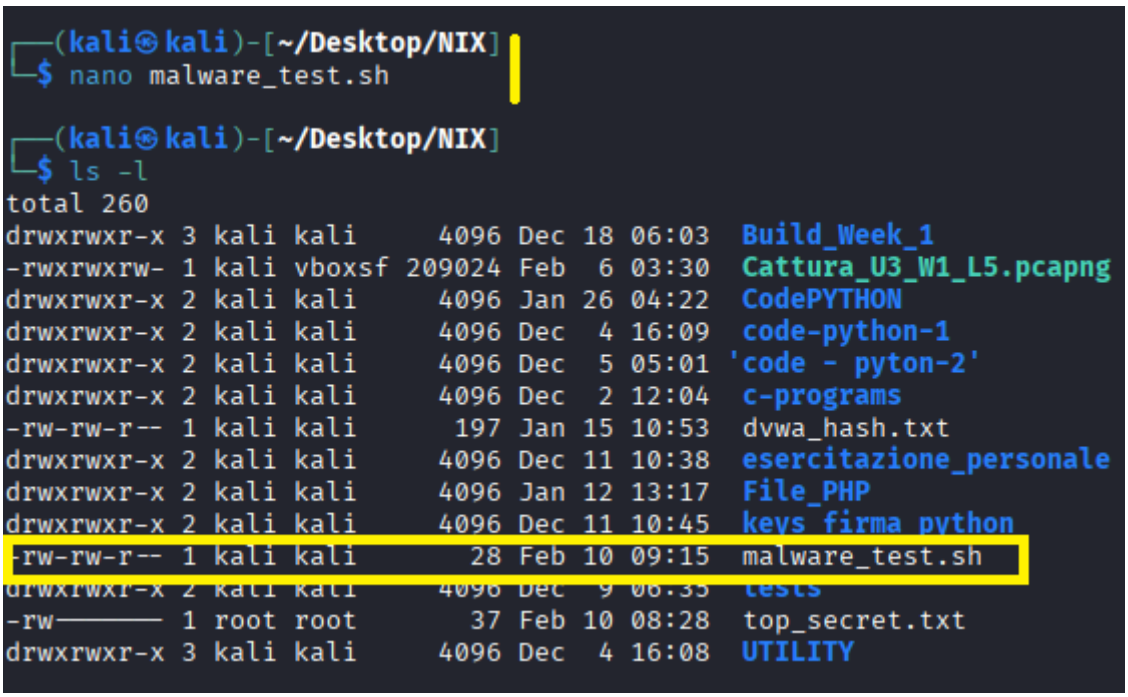

Fig. 4 Creazione script .sh

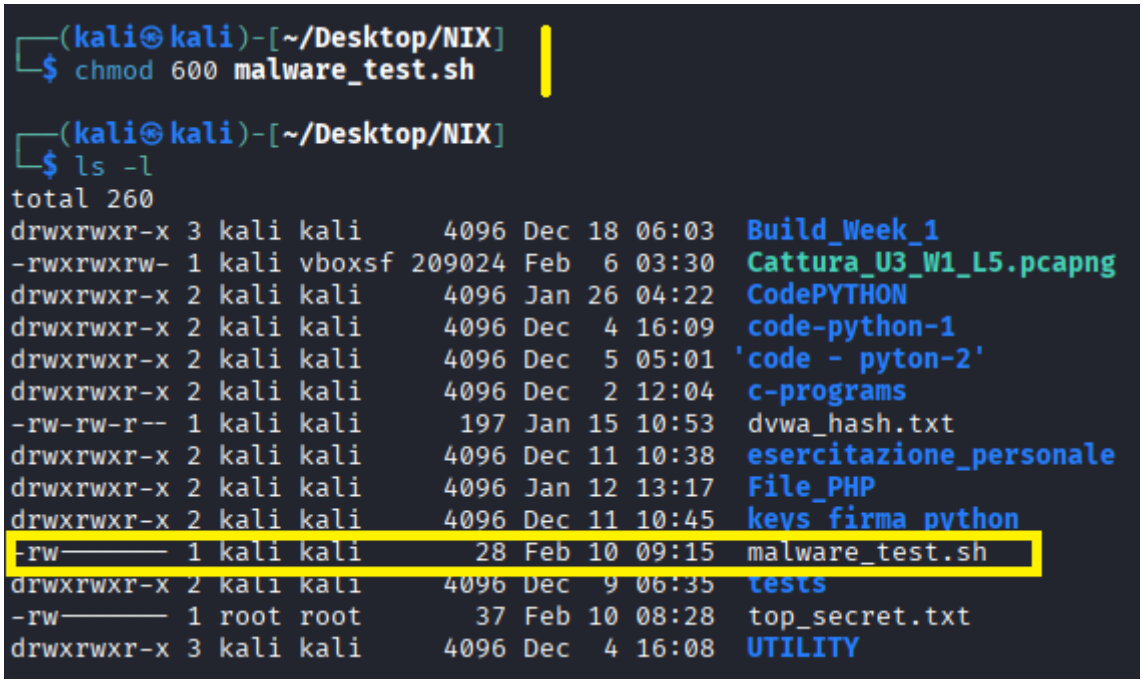

Fig. 5 Permission denied

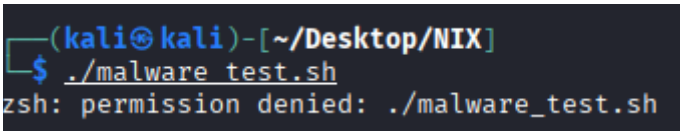

Fig. 6 Modifica permessi

6. Conclusioni

L'attività svolta ha evidenziato l'importanza cruciale della gestione dei **permessi** nei sistemi Linux.

Abbiamo osservato che i permessi di default (spesso **644** per i file) non sono sufficienti per proteggere dati sensibili, poiché consentono la lettura globale.

Attraverso l'uso del comando **`chmod`**, abbiamo applicato con successo una politica di restrizione, dimostrando che:

- È essenziale verificare sempre i permessi dopo la creazione di una risorsa.
- La corretta configurazione dei permessi è la prima linea di difesa contro accessi interni **non autorizzati** ed exfiltration di dati.

I file **`top_secret.txt`** e **`malware_test.sh`** sono ora sicuri e conformi agli standard di riservatezza richiesti.