Gestione dei permessi
S10L2

Traccia:

Abbiamo visto come si gestiscono i permessi in Linux.

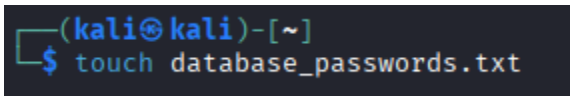
Obiettivo: Configurare e gestire i permessi di lettura, scrittura ed esecuzione per file o directory in un sistema Linux.

La scelta dei file o delle directory da configurare spetta allo studente.

Infine, lo studente dovrà creare degli screenshot che mostrino i passaggi effettuati e scrivere una relazione spiegando le scelte fatte riguardo ai permessi.

Fase 1: creazione del file

Ho creato un file denominato **`database_passwords.txt`** utilizzando il comando **`touch`**. Questo file simula un archivio di credenziali che non deve essere accessibile a utenti non autorizzati.



Fase 2: verifica dei permessi iniziali

Prima di applicare le policy di sicurezza, ho verificato i permessi di default assegnati dal sistema operativo tramite il comando **`ls -l`**.

Come si evince dall'output, il sistema ha assegnato permessi standard (**`rw-r--r--`**), che permetterebbero la lettura del file anche ad utenti del gruppo e ad altri utenti del sistema, rappresentando un rischio di sicurezza.

```
┌──(kali㉿kali)-[~]
└─$ ls -l database_passwords.txt
-rw-rw-r-- 1 kali kali 0 Nov 25 10:09 database_passwords.txt
```

Fase 3: modifica dei permessi

Per mettere in sicurezza il file, ho utilizzato il comando **chmod**. L'obiettivo era rendere il file leggibile **solo** dal proprietario (Owner) e non modificabile da nessuno (nemmeno accidentalmente dal proprietario stesso), rimuovendo ogni accesso per il Gruppo e gli Altri.

Comando eseguito: **chmod 400 database_passwords.txt**

```
┌──(kali㉿kali)-[~]
└─$ chmod 400 database_passwords.txt

┌──(kali㉿kali)-[~]
└─$ ls -l database_passwords.txt
-r-------- 1 kali kali 0 Nov 25 10:09 database_passwords.txt
```

Fase 4: test di validazione

Per verificare l'efficacia dei permessi configurati, ho tentato di scrivere all'interno del file utilizzando il comando **echo** con ridirezione **>>**.

Poiché ho rimosso il permesso di scrittura (**w**) anche per l'utente proprietario, il sistema ha bloccato l'operazione.

```
┌──(kali㉿kali)-[~]
└─$ echo "nuova password" >> database_passwords.txt
zsh: permission denied: database_passwords.txt
```

Conclusioni:

La scelta di impostare i permessi a `400` (`-r--------`) è basata sui seguenti criteri di sicurezza:

- **Lettura (read - r):** assegnata **esclusivamente** all'utente proprietario (`owner`). Questo garantisce la **confidenzialità**: nessun altro utente nel sistema (**group** o **others**) può visualizzare le password contenute nel file.
- **Scrittura (write - w):** rimossa per tutti, incluso il proprietario, questo garantisce l'**integrità**: il file non può essere modificato o cancellato accidentalmente. Se il proprietario dovesse aggiornare le password, dovrà esplicitamente elevare i propri privilegi temporaneamente, riducendo il rischio di errore umano o script malevoli.
- **Esecuzione (execute - x):** negata a tutti, trattandosi di un file di testo e non di uno script o un programma, il permesso di esecuzione non è necessario e rimuoverlo è una best practice per evitare che il file venga utilizzato come vettore di attacco.

Il test effettuato nella **fase 4** ha avuto esito positivo:
Il messaggio di errore **"permission denied"** restituito dalla shell conferma che il kernel Linux sta applicando correttamente le **access control list (ACL)** definite, questo dimostra che il file è protetto contro modifiche non autorizzate, soddisfacendo i requisiti di sicurezza ipotizzati nello scenario iniziale.