Progetto: Full Kill Chain su Linux (Metasploitable 2)

Autore: Elena Pagnacco **Data:** 21/01/2026

Sintesi

È stata condotta una simulazione di attacco completa (***Full Kill Chain***) su un sistema target *Linux* vulnerabile (Metasploitable 2). L'attività ha coperto tutte le fasi principali di un ***penetration test***: dall'accesso iniziale tramite un servizio mal configurato (***PostgreSQL***), all'escalation dei privilegi sfruttando una vulnerabilità del kernel, fino all'installazione di una persistenza per garantire l'accesso futuro. Il test ha dimostrato come la mancata applicazione di patch di sicurezza e configurazioni deboli possano portare alla totale compromissione del sistema.

Richieste

- Ottenere accesso iniziale (***Initial Access***) sfruttando il servizio ***PostgreSQL***.
- Eseguire la ***Privilege Escalation*** per passare da utente limitato a **Root**.
- Garantire la persistenza (***Persistence***) installando una ***backdoor SSH***.
- **Verificare l'accesso** come Root senza password.

Introduzione

Il presente documento riporta l'analisi tecnica di un'attività di ***Penetration Testing*** mirata a un server **Linux legacy**. Lo scenario simula un attaccante che, dopo aver identificato una porta aperta (5432/Postgres), tenta di sfruttarla per prendere il controllo della macchina. In questa sessione, l'escalation dei privilegi non è avvenuta tramite un exploit del ***kernel***, ma abusando di una funzionalità ***"feature-turned-vulnerability"*** del software di rete ***Nmap*** installato con privilegi amministrativi. L'attività è stata condotta in ambiente controllato (***Sandbox tra Kali Linux e Metasploitable 2***).

Svolgimento e Punti Chiave

1. Accesso Iniziale (Exploit PostgreSQL)

Dopo la fase di **ricognizione**, è stato identificato il servizio ***PostgreSQL*** attivo sulla porta **5432**. Utilizzando il framework ***Metasploit***, è stato lanciato l'exploit **linux/postgres/postgres_payload**. L'attacco ha avuto successo permettendo l'apertura di una sessione Meterpreter con i privilegi dell'utente di servizio postgres (uid=108).

- **Comando:** $ use exploit/linux/postgres/postgres_payload
- **Risultato:** Sessione 1 aperta (utente non privilegiato).

Figura 1 sessione 1 aperta, utente non privilegiato

2. Privilege Escalation (SUID Nmap)

Per ottenere i privilegi di ***Root***, è stata effettuata un'analisi dei binari presenti sul sistema tramite il modulo ***local_exploit_suggester***. È emerso che il tool **Nmap** (versione legacy) era installato con il bit **SUID** attivo. Le vecchie versioni di Nmap supportavano una "modalità interattiva" che permetteva di eseguire comandi shell. Poiché il binario aveva il permesso di essere eseguito come root (SUID), la shell generata ha ereditato tali privilegi.

- **Modulo utilizzato:** exploit/unix/local/setuid_nmap
- **Tecnica:** Abuso di binario **SUID** (Misconfigurazione).
- **Target:** Sessione 1 (Postgres).
- **Payload:** cmd/unix/reverse_netcat (utilizzato per aggirare problemi di architettura).
- **Risultato:** Ottenimento di una nuova shell con privilegi uid=0(root).

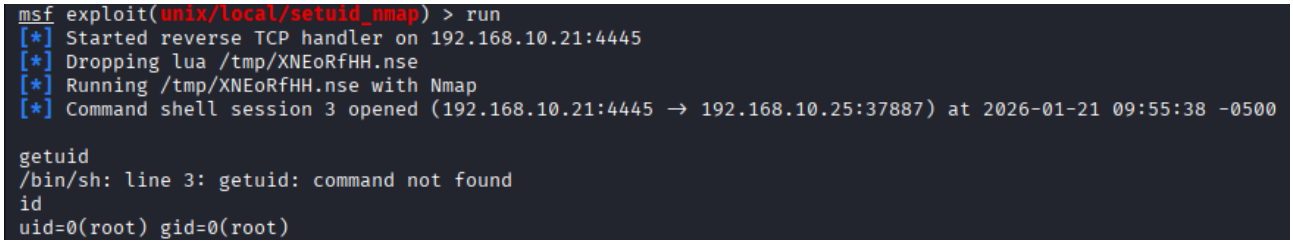

Figura 2 Sessione con privilegi di root

3. Persistenza (SSH Backdoor)

Una volta ottenuto l'accesso come Root, è stato necessario garantire la ri-connessione al sistema. È stato utilizzato il ***modulo di persistenza SSH*** che inietta una chiave pubblica malevola nel file ***authorized_keys*** dell'utente root.

- **Modulo utilizzato:** post/linux/manage/sshkey_persistence
- **Azione:** Generazione chiave RSA e iniezione nel file authorized_keys della vittima.

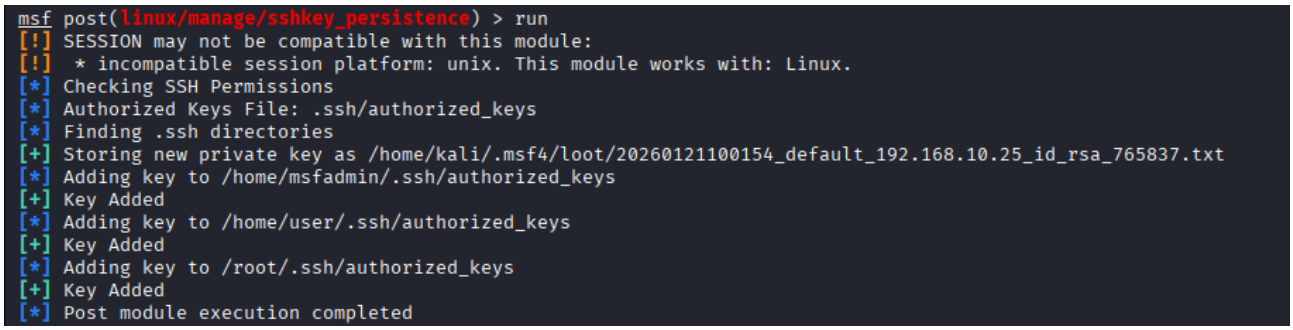

Figura 3 generazione e iniezione chiave SSH

4. Verifica dell'Accesso

Per confermare il successo dell'operazione, è stata simulata una connessione successiva dalla macchina attaccante (***Kali***) utilizzando la chiave privata esfiltrata. È stato necessario forzare l'uso di algoritmi legacy (ssh-rsa) sul client moderno per compatibilità con il server obsoleto.

- **Comando:** $ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i backdoor.key root@192.168.10.25
- **Risultato:** Accesso immediato al terminale di root senza password.

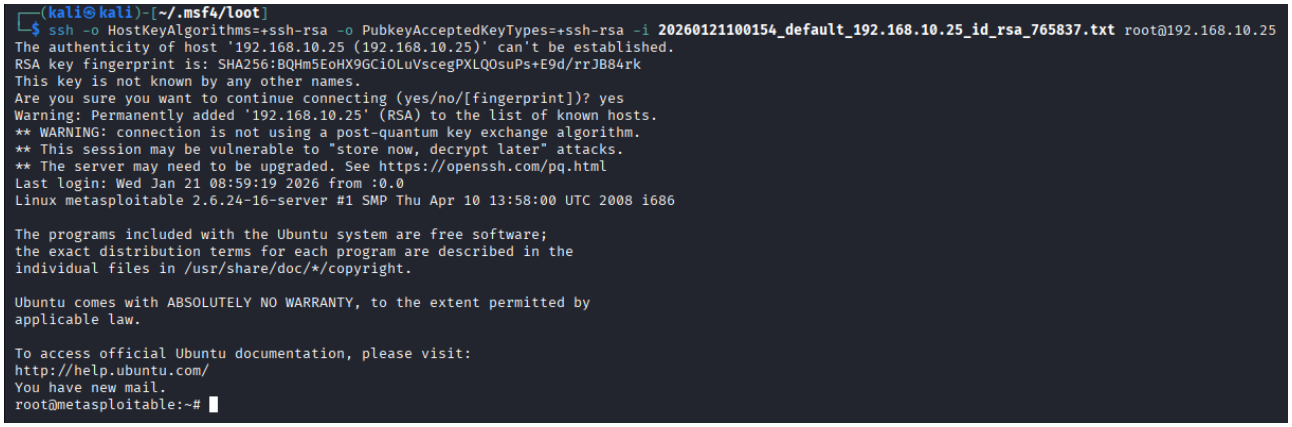

Figura 4 Accesso con chiave e uso di algoritmi legacy

Conclusioni e Mitigazioni

L'attacco ha avuto successo sfruttando una catena di configurazioni errate più che veri e propri bug software:

- **PostgreSQL Debole:** Il database permetteva l'esecuzione di comandi di sistema.
- **Permessi SUID Pericolosi:** La presenza del bit **SUID** su un software complesso come **Nmap** (chmod u+s /usr/bin/nmap) è una violazione critica delle best practice di sicurezza.
- **Software Obsoleto:** La versione di Nmap installata permetteva l'escape in shell (funzionalità rimossa nelle versioni moderne).

Azioni correttive raccomandate:

1. **Gestione Permessi (Least Privilege):** Rimuovere immediatamente il bit **SUID** dai binari che non ne hanno strettamente bisogno, specialmente editor di testo, interpreti e tool di rete.
    - **Comando fix***:* $ chmod u-s /usr/bin/nmap
2. **Aggiornamento Software:** Aggiornare ***Nmap*** a una versione recente che non supporti più la modalità interattiva.
3. **Hardening SSH:** Disabilitare l'accesso root via SSH e monitorare le modifiche ai file delle chiavi autorizzate.