PROGETTO METASPLOIT

Obiettivo dell'esercizio

In questo esercizio, l'obiettivo era sfruttare una vulnerabilità nel servizio **Java RMI** (Remote Method Invocation) sulla porta **1099** della macchina **Metasploitable** per ottenere una sessione **Meterpreter** sulla macchina vittima. Una volta ottenuta la sessione, dovevo raccogliere due informazioni dalla macchina vittima:

1. La configurazione di rete.
2. Le informazioni sulla tabella di routing.

Preparazione e configurazione iniziale

Per cominciare, ho configurato gli indirizzi IP delle due macchine come segue:

- **Macchina Kali (attaccante)**: 192.168.1.111
- **Macchina Metasploitable (vittima)**: 192.168.1.112

Successivamente, ho riavviato entrambe le macchine e ho verificato la connettività con un comando **ping**, che ha restituito un risultato positivo, confermando che le due macchine potevano comunicare correttamente tra di loro.

Scansione delle porte

Per verificare la presenza del servizio vulnerabile, ho eseguito una scansione con **Nmap** sulla porta **1099**, che è quella utilizzata dal servizio **Java RMI**. Il comando utilizzato è stato:

- **nmap -p 1099 192.168.11.112**

Il risultato ha confermato che la porta **1099** è aperta e in ascolto sul servizio **rmiregistry**.

Per simulare un attacco più realistico, ho poi eseguito una scansione meno aggressiva su tutte le porte della macchina vittima con il comando:

- **nmap -sS -Pn -T2 -p- 192.168.11.112**

Questo ha permesso di ottenere una panoramica delle porte aperte, senza generare troppo traffico sospetto.

Successivamente a ciò ho usato un'altro comando nmap **"--script=rmi-vuln-classloader -p 1099 192.168.11.112".**

Che verifica se il servizio **Java RMI** sulla porta **1099** della macchina **192.168.11.112** è vulnerabile a un attacco che permette il caricamento e l'esecuzione di classi maliziose.

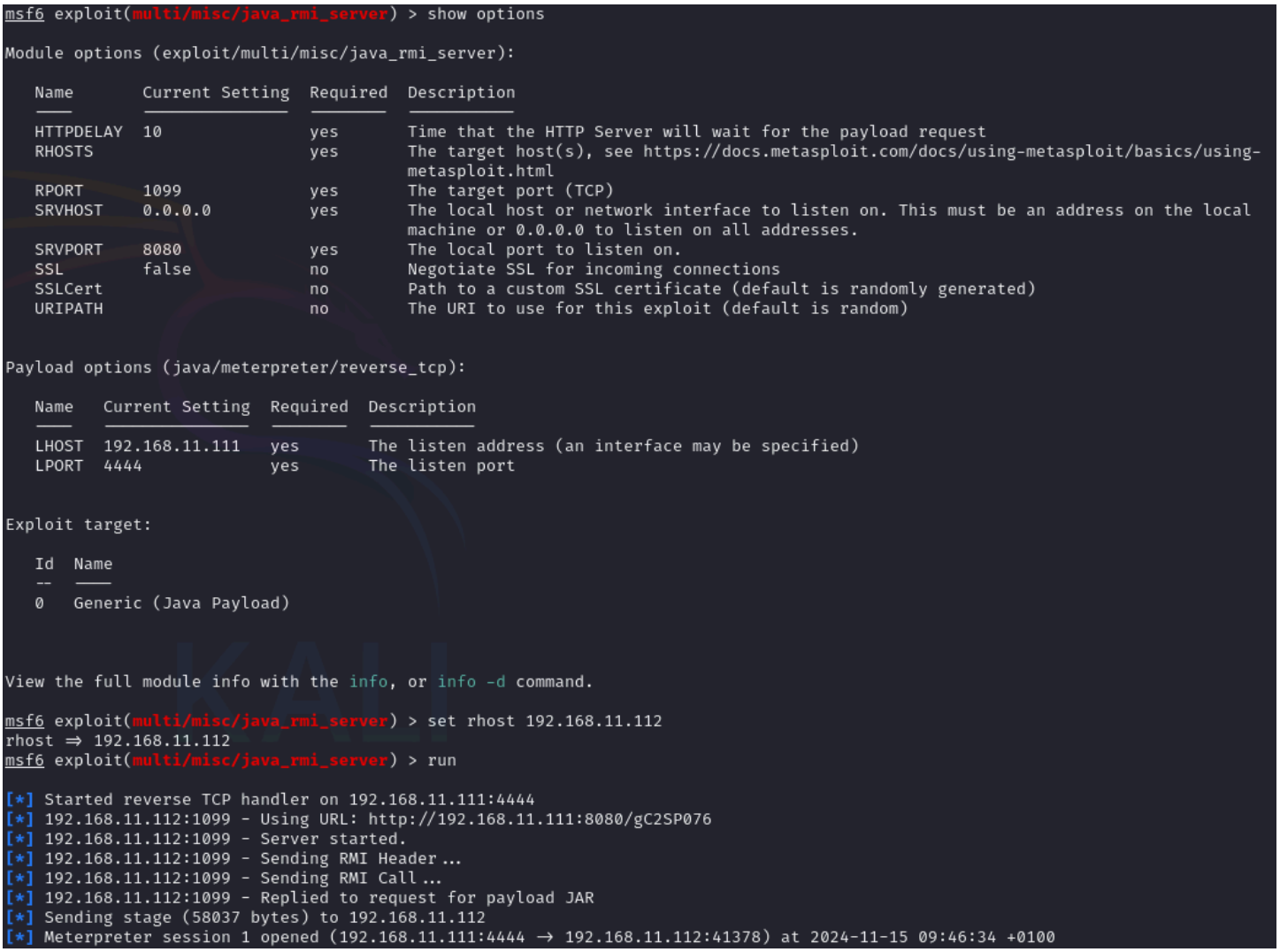
```
┌──(kali㉿kali)-[~]
└─$ nmap --script=rmi-vuln-classloader -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 10:54 CET
Nmap scan report for 192.168.11.112
Host is up (0.0011s latency).

PORT     STATE SERVICE
1099/tcp open  rmiregistry
| rmi-vuln-classloader:
|   VULNERABLE:
|   RMI registry default configuration remote code execution vulnerability
|     State: VULNERABLE
|       Default configuration of RMI registry allows loading classes from remot
e URLs which can lead to remote code execution.
|
|     References:
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/expl
oits/multi/misc/java_rmi_server.rb
MAC Address: 08:00:27:7E:B7:23 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

Sfruttamento della vulnerabilità con Metasploit

Dopo aver confermato la presenza del servizio vulnerabile, sono passato a **Metasploit**. Ho avviato la console di Metasploit (**msfconsole**) e ho cercato l'exploit per **Java RMI** con il comando **"search rmiregistry".**

```
msf6 > search  rmiregistry

Matching Modules
================

   #  Name                                     Disclosure Date  Rank       Check  Description
   -  ----                                     ---------------  ----       -----  -----------
   0  exploit/multi/misc/java_rmi_server       2011-10-15       excellent  Yes    Java RMI Server Insecure Default Configuratio
n Java Code Execution
   1    \_ target: Generic (Java Payload)      .                .          .      .
   2    \_ target: Windows x86 (Native Payload)  .              .          .      .
   3    \_ target: Linux x86 (Native Payload)  .                .          .      .
   4    \_ target: Mac OS X PPC (Native Payload)  .             .          .      .
   5    \_ target: Mac OS X x86 (Native Payload)  .             .          .      .
```

Il primo exploit trovato è stato **exploit/multi/misc/java_rmi_server**, che ho selezionato con **"use".**



Ho quindi configurato l'indirizzo IP della macchina vittima e successivamente, ho avviato l'exploit con il comando.

A questo punto, è stata creata una sessione **Meterpreter**, che mi ha permesso di interagire con la macchina vittima.

Raccolta delle Info

Una volta ottenuta la sessione Meterpreter, ho eseguito i seguenti comandi per raccogliere le informazioni richieste:

Configurazione di rete: Ho usato il comando:

- **ifconfig**

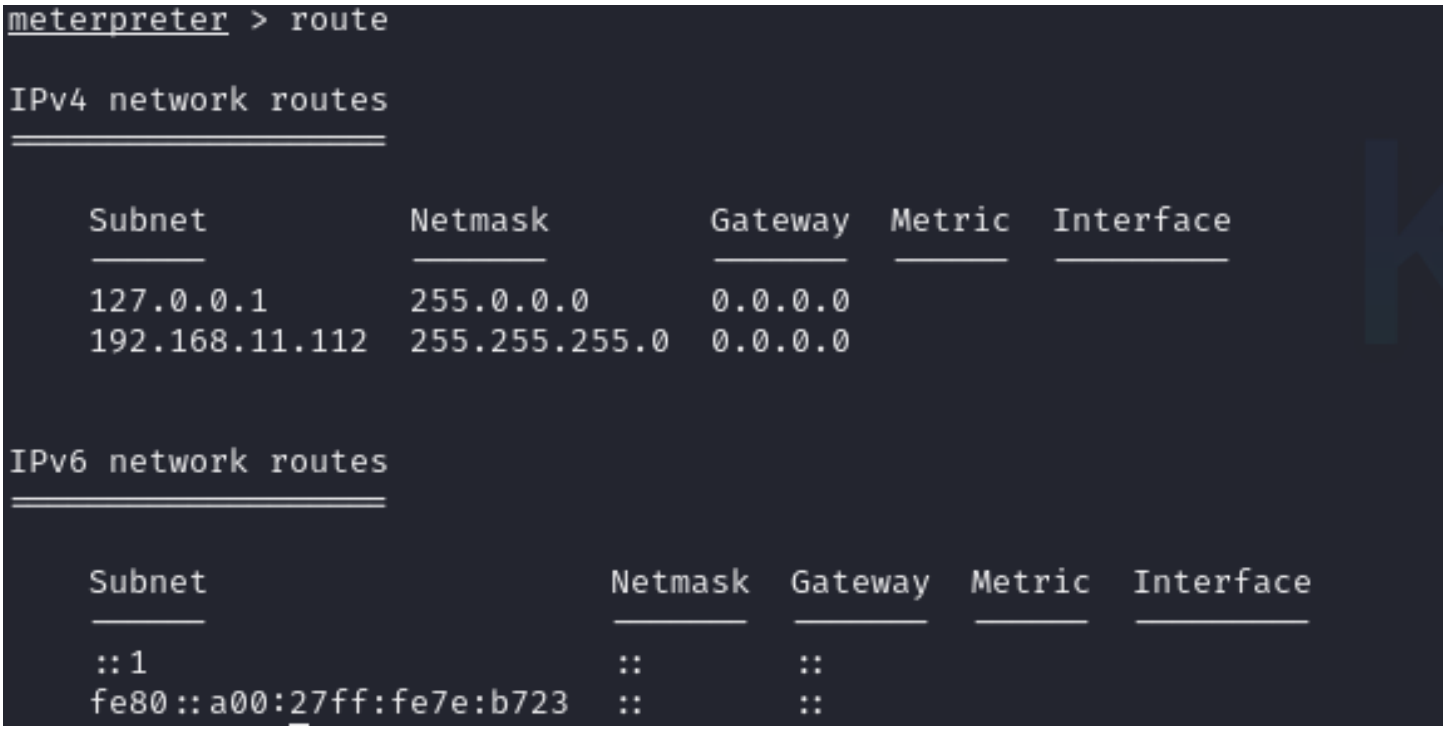
Questo comando ha restituito gli indirizzi IP, la subnet mask e il gateway della macchina vittima.

```
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:fe7e:b723
IPv6 Netmask : ::
```

Per la tabella di routing ho eseguito:

- **route**

Questo ha mostrato la tabella di routing, che indica come la macchina vittima gestisce il traffico di rete e a quali destinazioni invia i pacchetti.



Errore HTTPDELAY e risoluzione

Durante l'esecuzione dell'exploit, potrebbe verificarsi un errore chiamato **HTTPDELAY**. Questo errore si verifica quando ci sono ritardi nelle comunicazioni tra il client Metasploit e la macchina vittima, causando una connessione instabile.

In caso di questo errore, è necessario configurare il parametro **HTTPDELAY** a un valore maggiore, come **20**, per ridurre il ritardo nella comunicazione. Il comando per farlo è:

set HTTPDELAY 20

Perché è stato usato questo exploit

L'exploit **java_rmi_server** è stato scelto perché sfrutta una vulnerabilità nel servizio **Java RMI Registry**, che consente a un attaccante di inviare richieste remote malformate al server. Questo può portare all'esecuzione di codice arbitrario sulla macchina vittima, come nel caso in cui otteniamo una sessione **Meterpreter**.

Il servizio **Java RMI** non verifica adeguatamente le richieste di invocazione, e questo lo rende vulnerabile a exploit come quello che abbiamo utilizzato in questo esercizio.

Come risolvere la vulnerabilità Java RMI

Per mitigare la vulnerabilità legata a **Java RMI**, è importante prendere alcune misure di sicurezza:

1. **Aggiornamenti regolari**: Assicurarsi che il software Java sia sempre aggiornato per applicare le patch di sicurezza che risolvono le vulnerabilità note.
2. **Autenticazione e cifratura**: Configurare il servizio Java RMI per richiedere l'autenticazione e cifrare le comunicazioni, in modo da proteggere le invocazioni remote da attacchi di tipo **man-in-the-middle**.
3. **Limitare l'accesso alla porta 1099**: Utilizzare un firewall per limitare l'accesso alla porta **1099** (Java RMI Registry) solo ai client autorizzati.
4. **Disabilitare Java RMI se non necessario**: Se il servizio RMI non è indispensabile, è consigliabile disabilitarlo per ridurre il rischio di attacchi.

Conclusione

In questo esercizio ho sfruttato una vulnerabilità nel servizio **Java RMI** di Metasploitable per ottenere una sessione **Meterpreter** e raccogliere informazioni sulla configurazione di rete e sulla tabella di routing della macchina vittima. L'exploit **java_rmi_server** è stato scelto perché sfrutta la vulnerabilità specifica di **Java RMI Registry**.