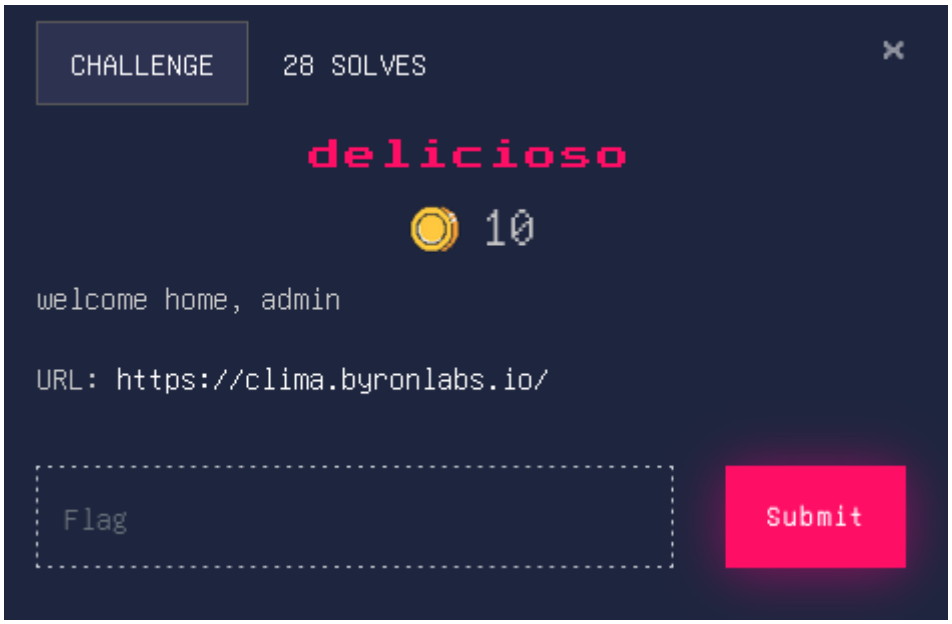
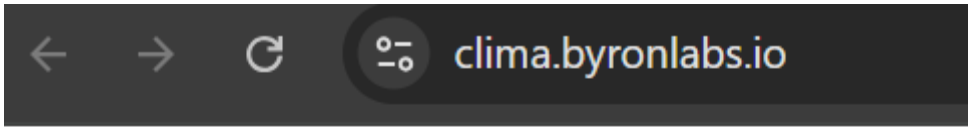
RETOS PENTESTING

RETO 1





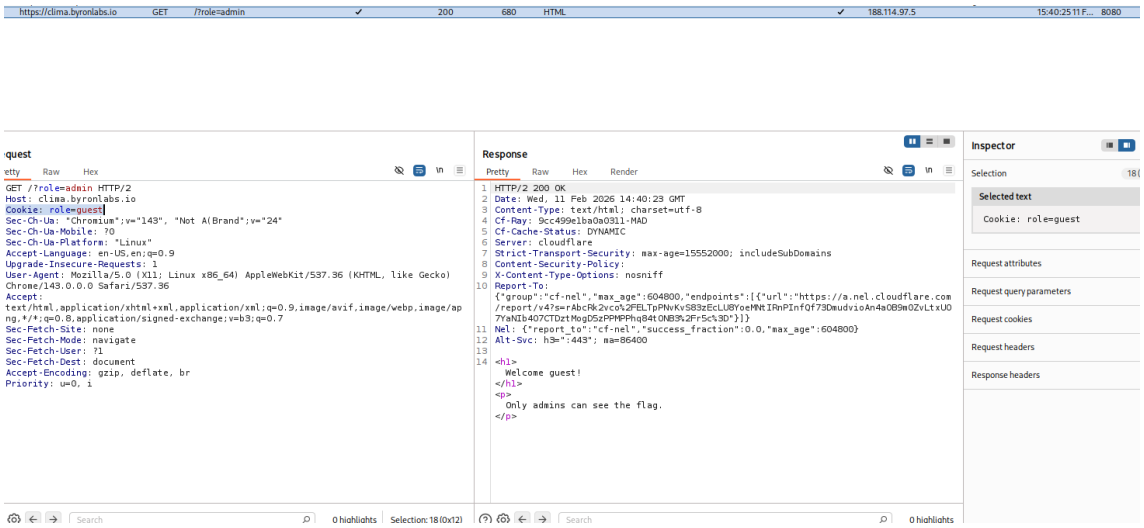
Welcome Guest!

Only admins can see the flag.

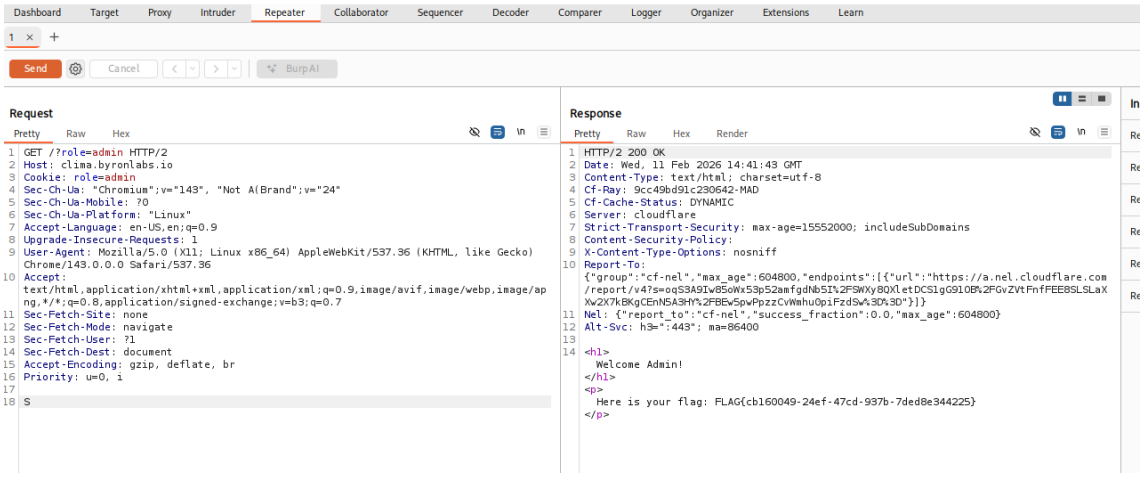
Abrimos burpsuite para ver la petición

Como no observamos nada en la primera petición vamos a probar a poner un query string con un role=admin

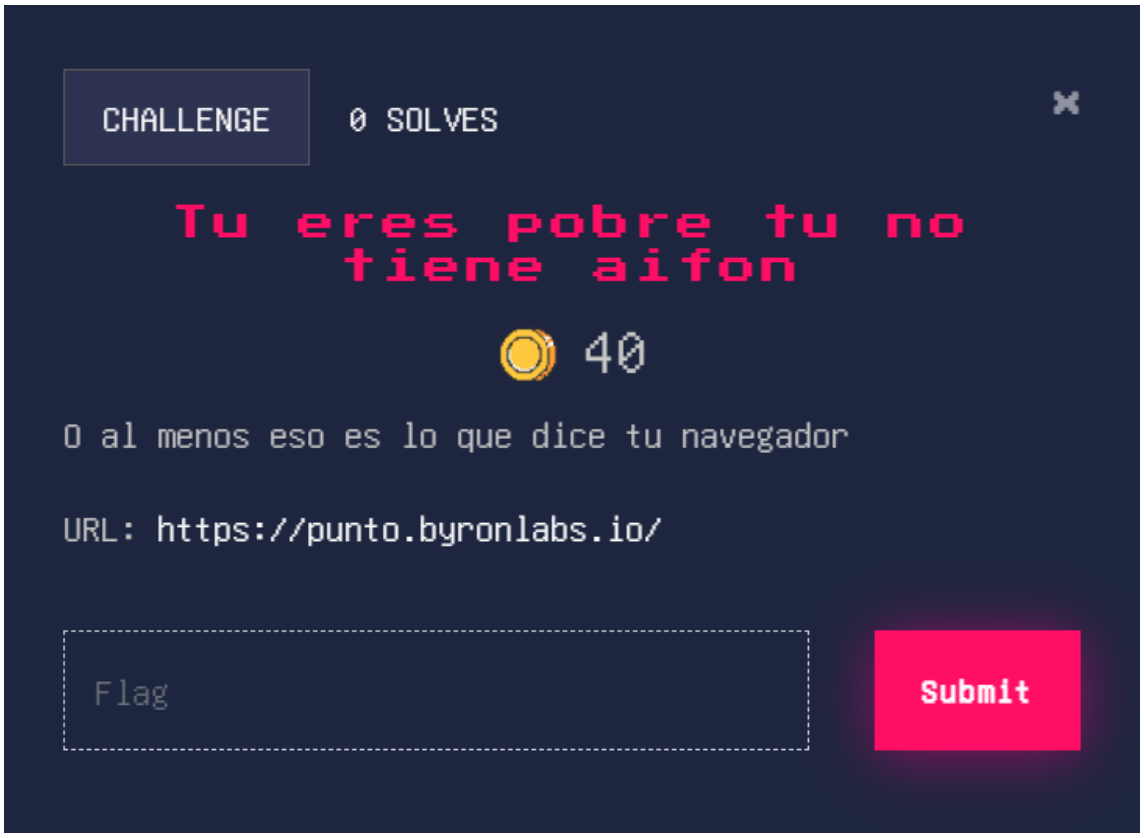
Y lanzando esta petición vemos en burpsuite que se nos ha asignado una cookie role=guest así que vamos a lanzar la petición al repeater y a manipularla

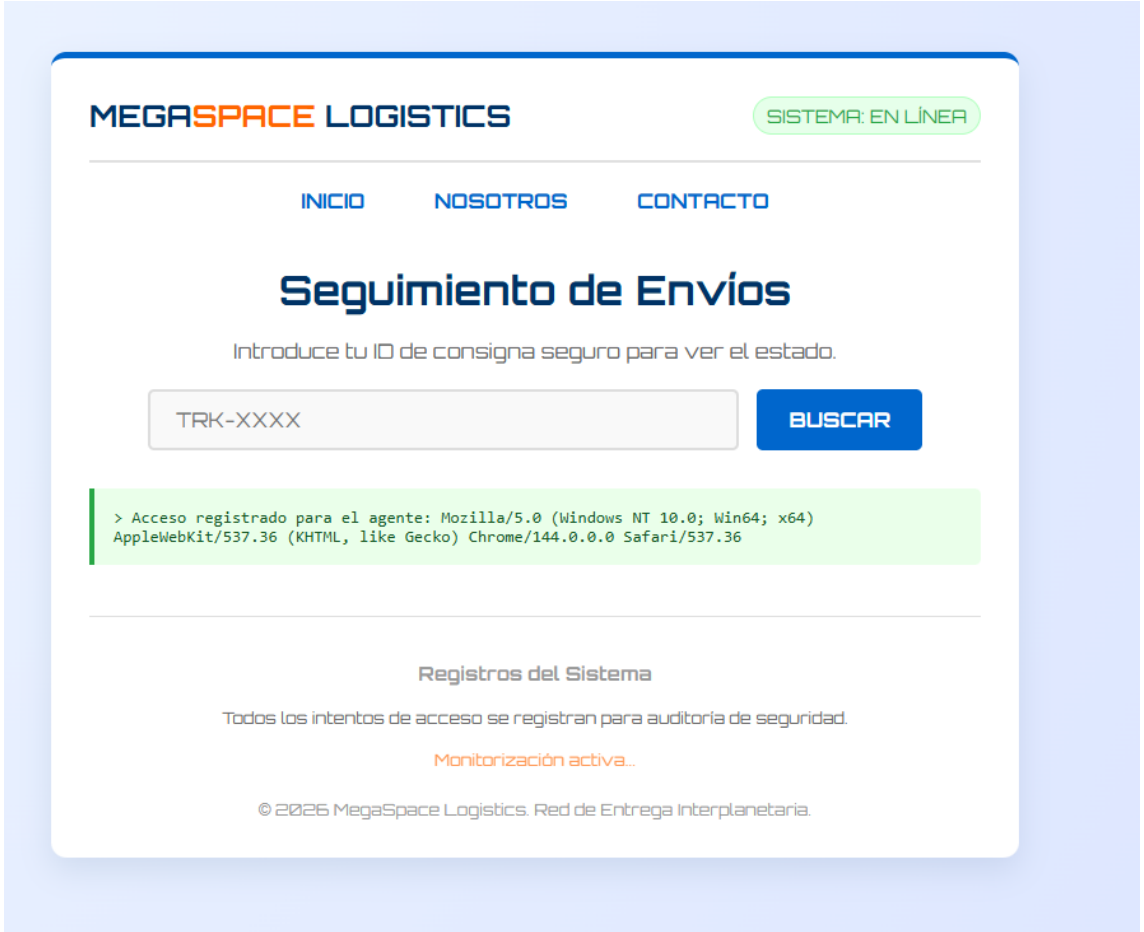


Y así es como obtenemos nuestra flag



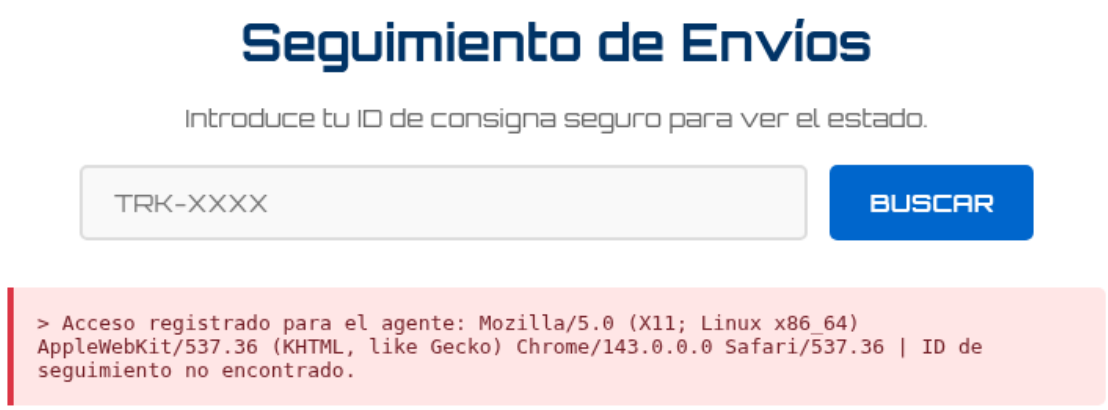
Reto 2


CHALLENGE
0 SOLVES
Tu eres pobre tu no tiene aifon
40
O al menos eso es lo que dice tu navegador
URL: https://punto.byronlabs.io/
Flag
Submit


MEGASPACE LOGISTICS
SISTEMA: EN LÍNEA
INICIO
NOSOTROS
CONTACTO
Seguimiento de Envíos
Introduce tu ID de consigna seguro para ver el estado.
TRK-XXXX
BUSCAR
> Acceso registrado para el agente: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Registros del Sistema
Todos los intentos de acceso se registran para auditoría de seguridad.
Monitorización activa...
© 2026 MegaSpace Logistics. Red de Entrega Interplanetaria.

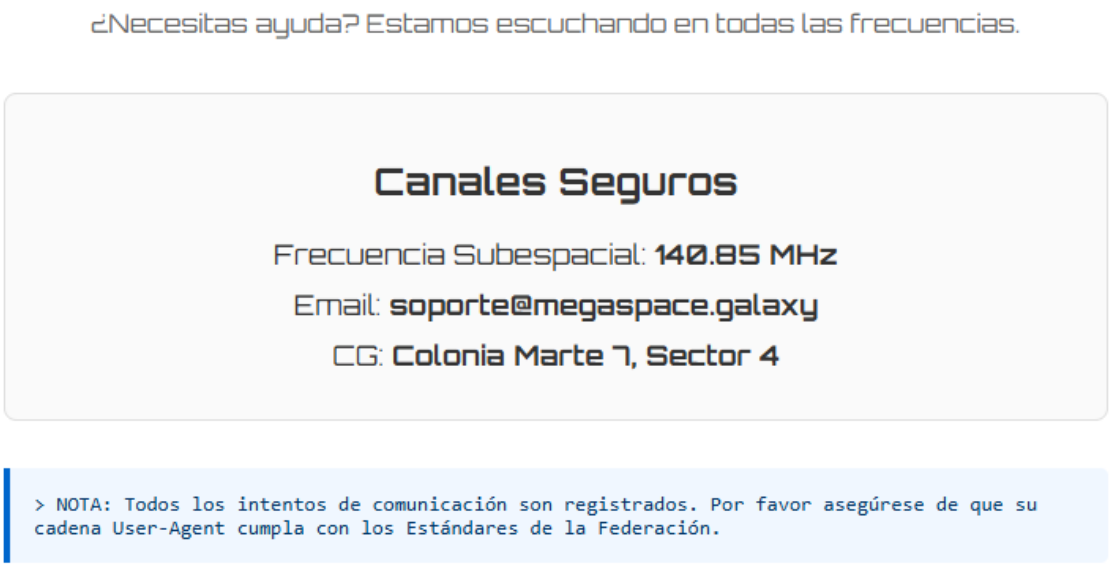
Observamos la pagina.

Vemos un placeholder en el buscador de TRK-XXXX por lo que pinta que el ID será del estilo TRK-FLAG



Probamos como no obtenemos nada vamos a seguir observando la pagina

Encontramos lo siguiente:

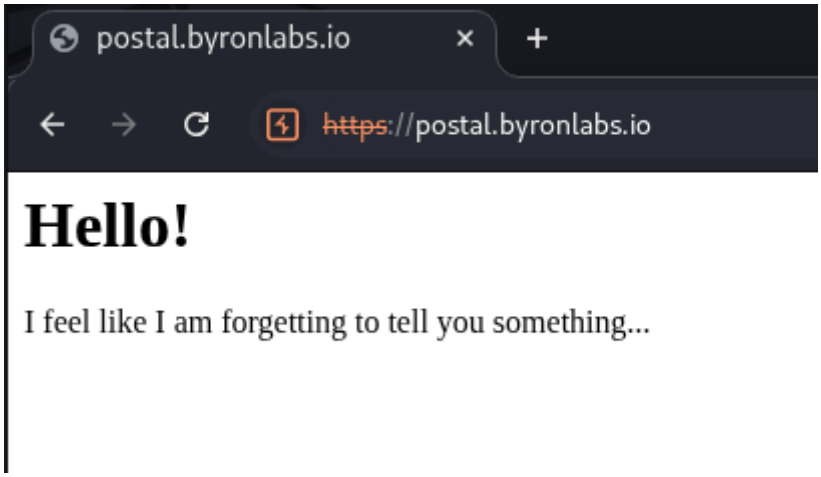


Por lo que seguramente tengamos que modificar el User-Agent

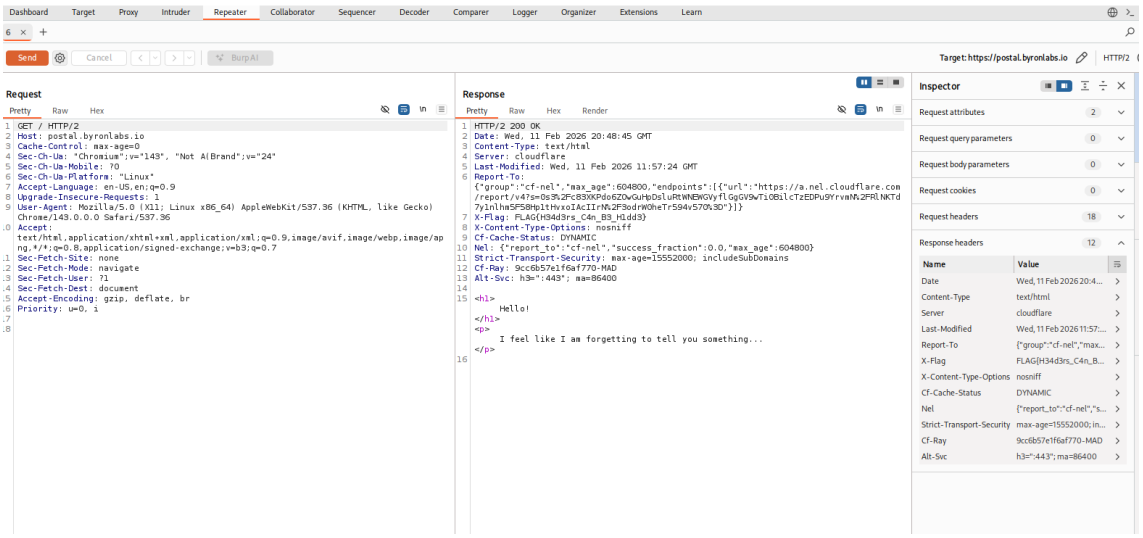
Con algo relacionado de esta información, quizás la frecuencia sea la versión del Chrome

RETO 3

Interceptamos la petición con BurpSuite

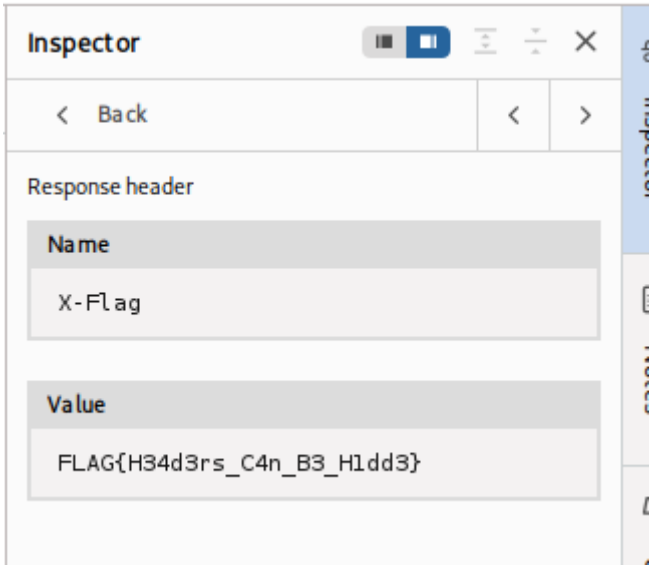


Y al final buscando e investigando la petición hemos encontrado que el servidor nos devuelve en los headers de la respuesta una cabecera X-Flag con la flag


Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Send
Cancel
Target: https://postal.byronlabs.io
HTTP/2
Request
Pretty
Raw
Hex
GET / HTTP/2
Host: postal.byronlabs.io
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
Response
Pretty
Raw
Hex
Render
HTTP/2 200 OK
Date: Wed, 11 Feb 2026 20:48:45 GMT
Content-Type: text/html
Server: cloudflare
Last-Modified: Wed, 11 Feb 2026 11:57:24 GMT
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=0s3%2Fc83XKPdo6ZOwGuHpDsluRtWNEWGVyflGgGV9wTiOBilcTzEDPu9YrvmN%2FRlNKTd7y1nlhm5F5BHp1tHvxoIAcIIrM%2F3odrWOheTr594v570%3D"}]}
X-Flag: FLAG{H34d3rs_C4n_B3_H1dd3}
X-Content-Type-Options: nosniff
Cf-Cache-Status: DYNAMIC
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Strict-Transport-Security: max-age=15552000; includeSubDomains
Cf-Ray: 9cc6b57e1f6af770-MAD
Alt-Svc: h3=":443"; ma=86400
<h1>
Hello!
</h1>

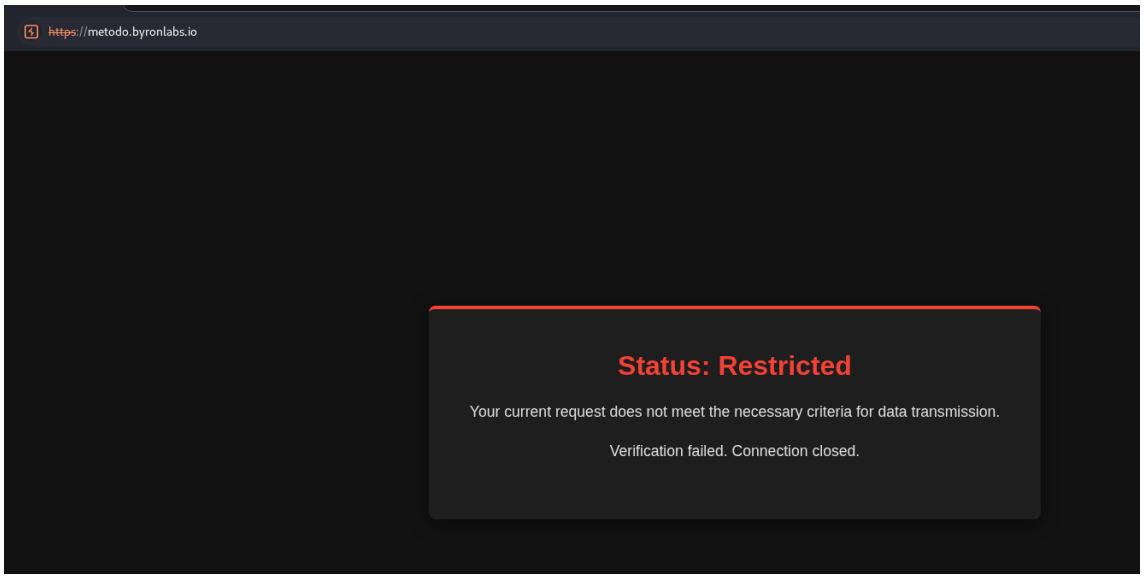
I feel like I am forgetting to tell you something...

Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Response headers
Name
Value
Date
Content-Type
Server
Last-Modified
Report-To
X-Flag
X-Content-Type-Options
Cf-Cache-Status
Nel
Strict-Transport-Security
Cf-Ray
Alt-Svc


Inspector
Back
Response header
Name
X-Flag
Value
FLAG{H34d3rs_C4n_B3_H1dd3}

Reto 4

Nos dan una pista en el titulo referente al method que hay que usar en la request



Así que vamos a observar la request desde burpsuite a ver si hay algo de info

Lanzando la petición GET no vemos nada así que vamos a usar el método OPTIONS para ver que métodos están permitidos

Request

Pretty Raw Hex

```
OPTIONS / HTTP/2
Host: metodo.byronlabs.io
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Date: Wed, 11 Feb 2026 17:53:41 GMT
Content-Type: text/html; charset=utf-8
Server: cloudflare
Allow: HEAD, OPTIONS, GET, POST
Cf-Cache-Status: DYNAMIC
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Content-Type-Options: nosniff
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com
/report/v4?s=83adaVO8GnJtSHfCUNtnYVA%2FP6aWJkH14mubA51qpuqOjBqJ496Fgwbs70PE1BNVVFfAK
Wtv4ftdGQNXmnTlGBWDTBtmsZ4jcYKDK%2Bby4mP1RcO%3D"}]}
Cf-Ray: 9cc5b50a6dda0355-MAD
Alt-Svc: h3=":443"; ma=86400
```

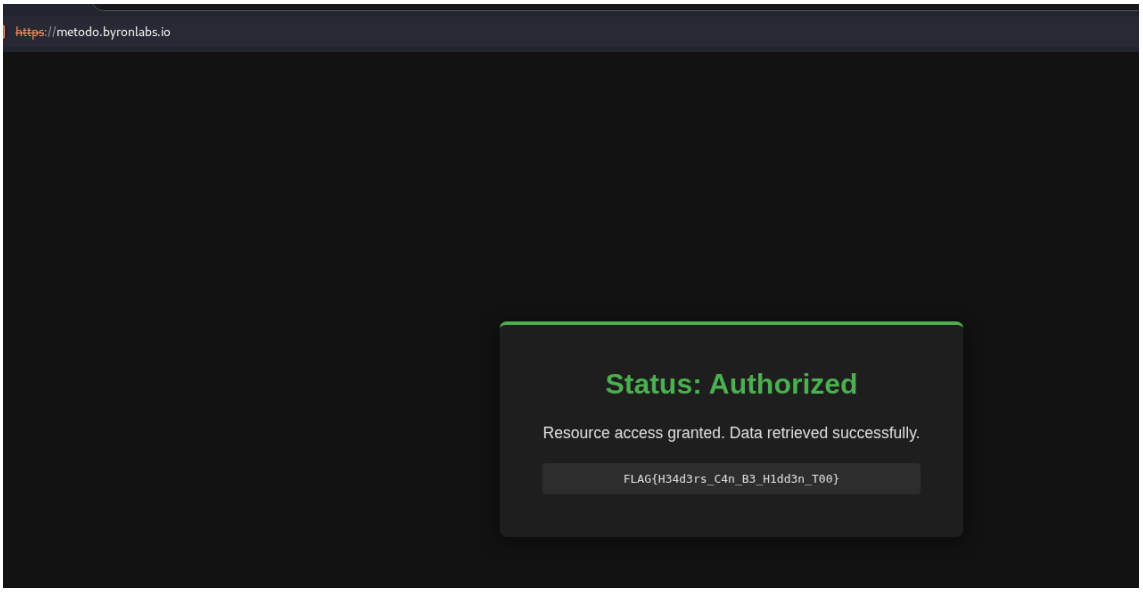
Los permitidos son HEAD, OPTIONS, GET Y POST. Asi que solo nos falta probar el POST

Request

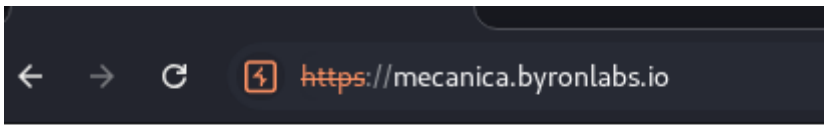
Pretty Raw Hex

```
POST / HTTP/2
Host: metodo.byronlabs.io
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Y Al editar la petición con el método POST obtenemos la flag

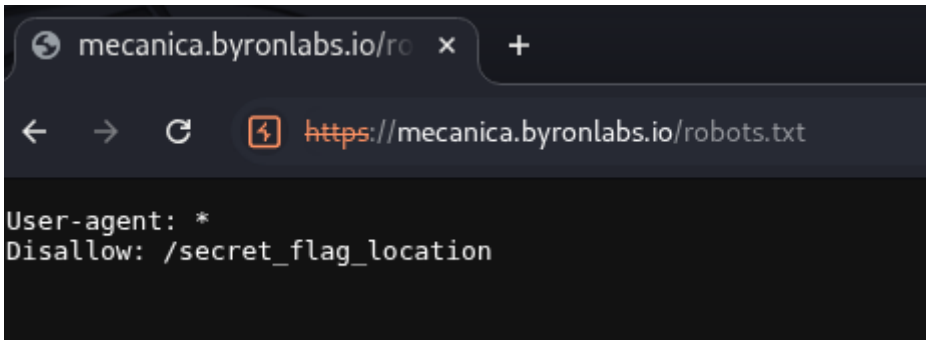


RETO 5

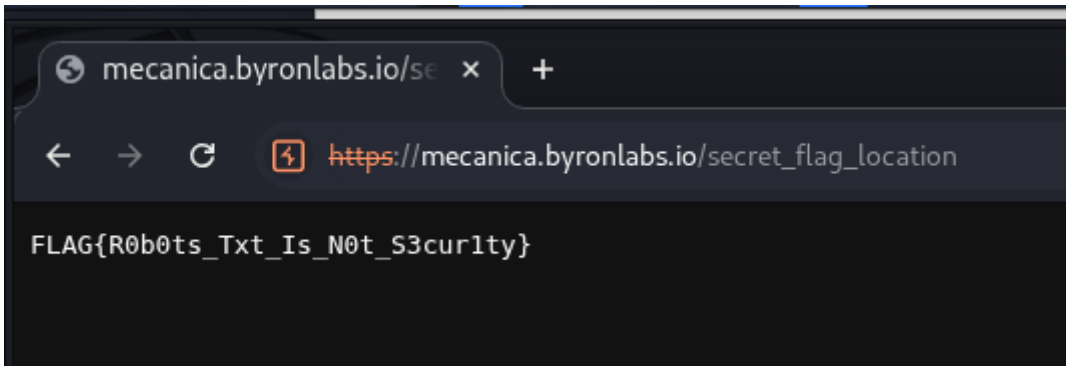


Welcome to my website!

Miramos el robots.txt

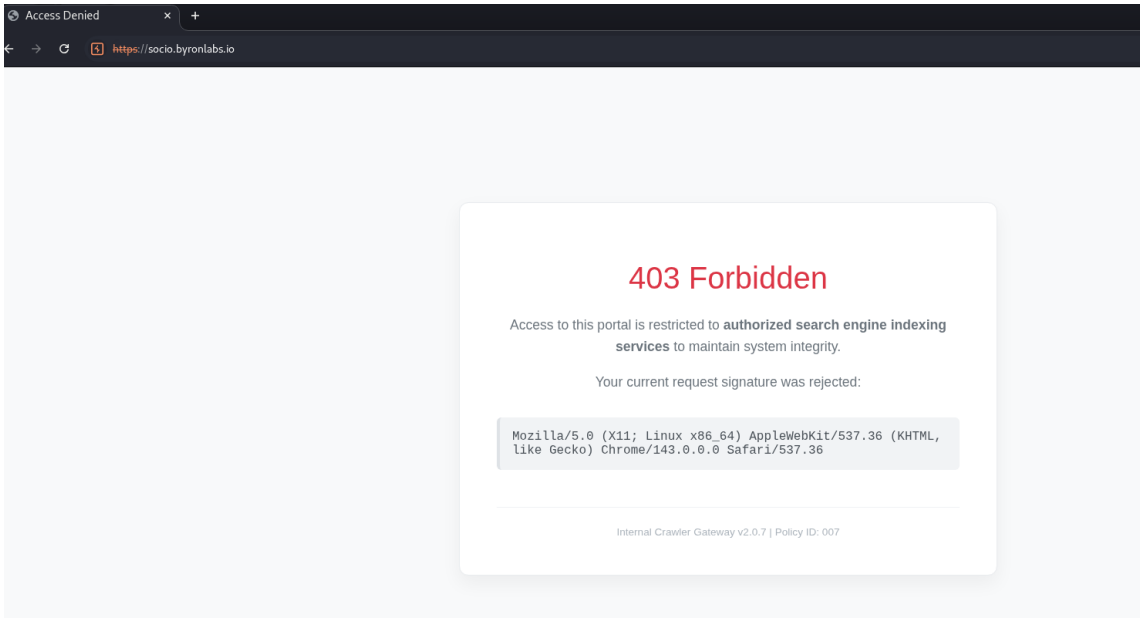


Y obtenemos nuestra flag


mecanica.byronlabs.io/se
https://mecanica.byronlabs.io/secret_flag_location
FLAG{R0b0ts_Txt_Is_N0t_S3cur1ty}

RETO 6

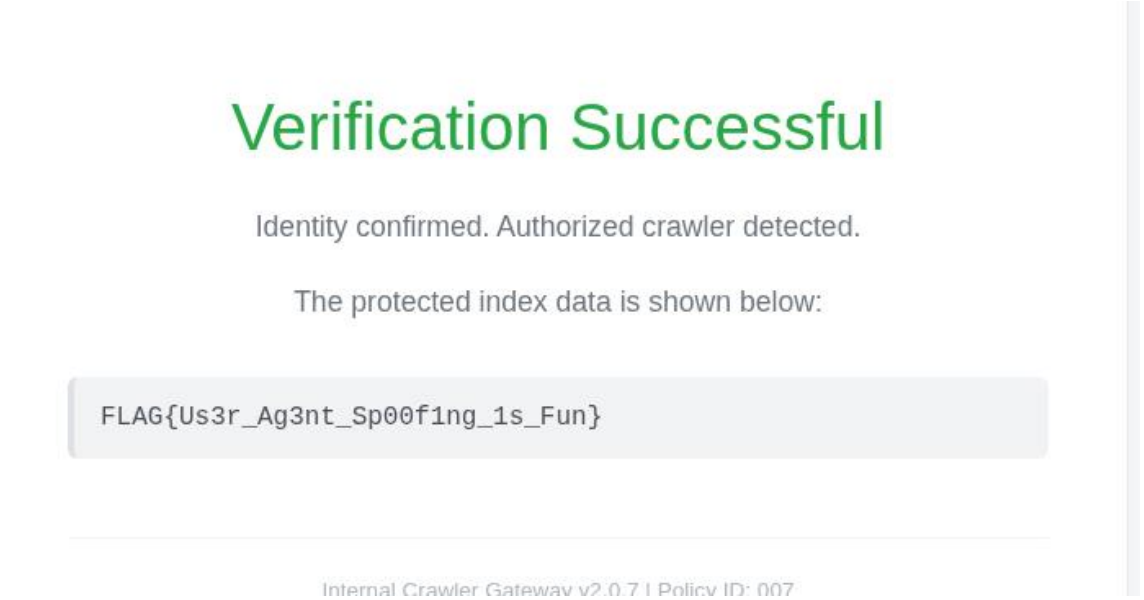

CHALLENGE
70 SOLVES
ok google
100
¿eres humano?
URL: https://socio.byronlabs.io/
Flag
Submit


Access Denied
https://socio.byronlabs.io
403 Forbidden
Access to this portal is restricted to authorized search engine indexing services to maintain system integrity.
Your current request signature was rejected:
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Internal Crawler Gateway v2.0.7 | Policy ID: 007

Según vemos en la pagina esta restringido solo por autorizados search engine indexing service así que debemos modificar el user-agent y hacernos pasar por un scrapper valido

Así que modificamos le user-agent por Googlebot (podemos usar típicos pero tiro a lo fácil ya que en el titulo pone ok Google nos decantamos por este)

Y obtenemos la flag



Reto 7

Page Viewer

Home | About

Select a page.

Revisando la página obtenemos esto así que probablemente haya local file inclsion LFI

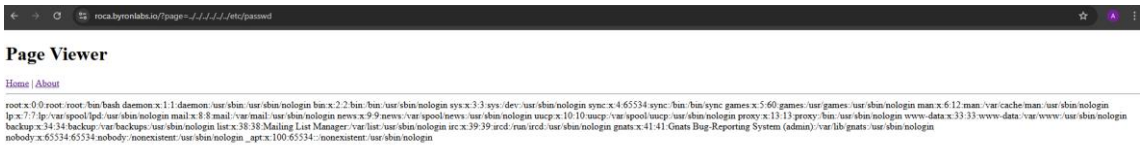
Page Viewer

Home | About

Warning: include(about.php): failed to open stream: No such file or directory in **/var/www/html/index.php** on line **17**

Warning: include(): Failed opening 'about.php' for inclusion (include_path='.:/usr/local/lib/php') in **/var/www/html/index.php** on line **17**

Esto lo verificamos con el /etc/passwd



Por más que buscamos un archivo flag.txt no encontramos nada, así que lo único que nos queda es leer el index.php porque si lo lanzamos se nos genera un bucle y peta.

Para ello usamos lo siguiente:
php://filter/convert.base64-encode/resource=index.php

Este wrapper codifica el contenido del archivo en Base64, permitiendo visualizar el código fuente sin que PHP lo procese.

Copiando el código fuente del index.php lo decodificamos
PD9waHANCi8vIGNvbmZpZy5waHAgY29udGFpbnMgdGhlIHNlY3JldCwgYnV0IHZpc2l0aW5nIGl0IGRpcmVjdGx5IHNob3dzIG5vdGhpbmcgKFBIUCBleGVjdXRlZCkNCiRmbGFnID0gIkZMQUd7TEZJX1cxdGhfUEhQX1dyYXBwM3JzfSI7DQo/Pg0KPCFET0NUWVBFIGh0bWw+DQo8aHRtbD4NCg0KPGgx PlBhZ2UgVmlld2VyPC9oMT4NCiAgICA8YSBocmVmPSI/cGFnZT1ob21lLnBocCI +SG9tZTwvYT4gfCA8YSBocmVmPSI/cGFnZT1hYm91dC5waHAiPkFib3V0PC9h Pg0KICAgIDxocj4NCiAgICA8P3BocA0KICAgICRwYWdlID0gJF9HRVRbJ3BhZ2U nXTsNCiAgICBpZiAoaXNzZXQoJHBhZ2UpKSB7DQogICAgICAgIC8vIFZVTE5FUk FCTEU6IERpcmVjdCBpbmNsdWRlIG9mIHVzZXIgaW5wdXQNCiAgICAgICAgLy8gSGFyZGVuaW5nOiBXZSByZXN0cmljdCB0byBjdXJyZW50IGRpcmVjdG9yeSBz bGlnaHRseSBidXQgYWxsb3d3cmFwcGVycw0KICAgICAgICBpbmNsdWRlKCR wYWdlKTsNCiAgICB9IGVsc2Ugew0KICAgICAgICBlY2hvICJTZWxlY3QgYSBwY WdlLiI7DQogICAgfQ0KICAgID8+DQoNCjwvaHRtbD4=

Y obtenemos la flag

BidXQgYWxsb3d3cmFwcGVycw0KICAgICAgICBpbmNsdWRlKCRwYWdlKTsNCiAgICB9IGVsc2Ugew0KICAgICAgICBlY2hvICJTZWxlY3QgYSBwYWdlLiI7DQogICAgfQ0KICAgID8+DQo8L2JvZHk+DQo8L2h0bWw+

Para binarios codificados (como imágenes, documentos, etc.) utilice el formulario de carga de archivos que encontrará un poco más abajo en esta página.

ASCII Conjunto de caracteres de origen.

Decodifique cada línea por separado (útil cuando tiene varias entradas).

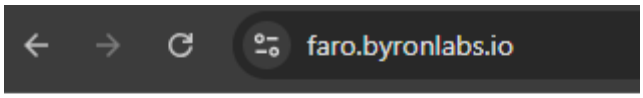
Modo en directo DESACTIVADO Decodifica en tiempo real mientras escribe o pega (sólo admite el juego de caracteres UTF-8).

DECODIFICAR Decodifica sus datos en la zona de abajo.

```
<?php
// config.php contains the secret, but visiting it directly shows nothing (PHP executed)
$flag = "FLAG{LFI_W1th_PHP_Wrapp3rs}";
?>
<!DOCTYPE html>
<html>

<body>
    <h1>Page Viewer</h1>
    <a href="?page=home.php">Home</a> | <a href="?page=about.php">About</a>
    <hr>
    <?php
```

Copiar al portapapeles

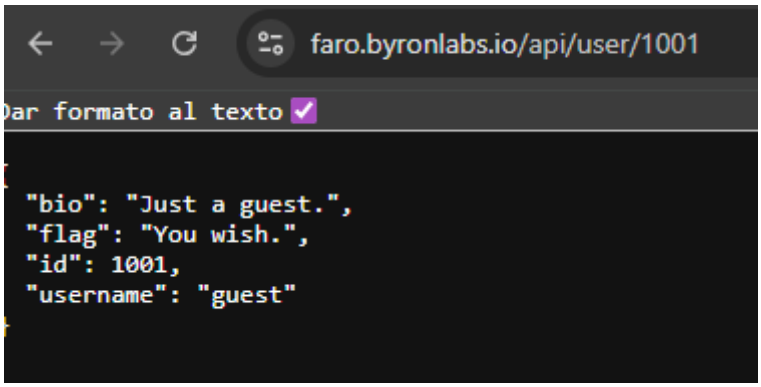
RETO 8



API Profile Viewer

Check your profile at /api/user/1001

checkeamos le profile



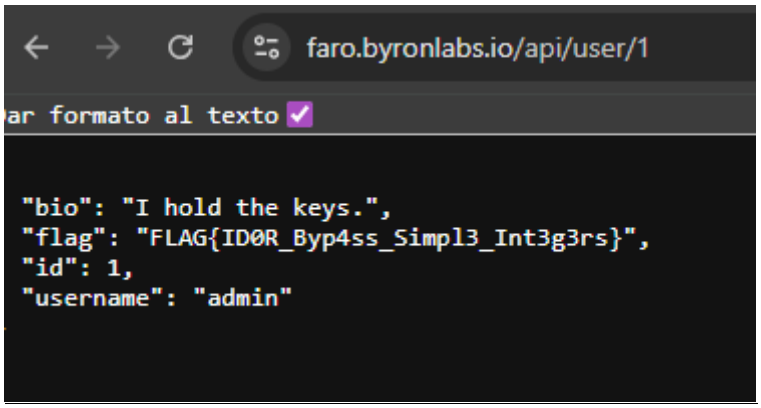
Y vemos que el endpoint es /api/users/{id}

Por lo que solo nos queda probar números

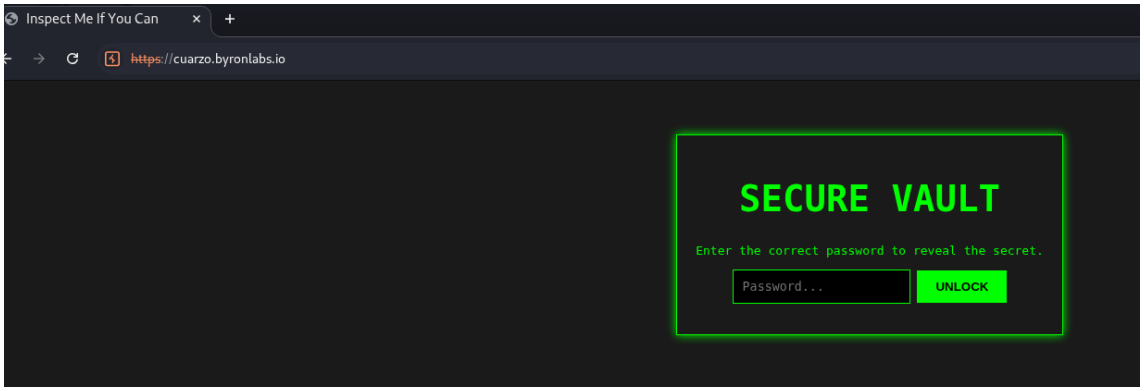
RETO 9

Porbamos el 1000 no existe, probamos el 1002

Probamos el id = 1 y bingo ^^ que es el usuario admin



RETO 10

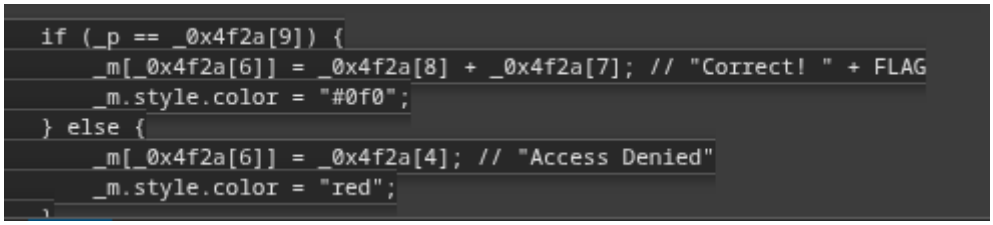


Inspeccionamos la pagina y vemos un script el cual leemos y vemos lo siguiguiente:

Logic: Compare input with hardcoded string (obfuscated in array)

así que en la variable ofuscada esta la contraseña o la flag, sabemos que los primeros valores son: 0: value, 1: getElementById, 2: pass, 3: length, 4: Access Denied, 5: msg, 6: innerText porque nos lo dicen en un comentario del código.
lógica ofusada

Si seguimos leyendo nos damos cuenta que el elemento 9 del array es la contraseña



```
if (_p == _0x4f2a[9]) {
    _m[_0x4f2a[6]] = _0x4f2a[8] + _0x4f2a[7]; // "Correct! " + FLAG
    _m.style.color = "#0f0";
} else {
    _m[_0x4f2a[6]] = _0x4f2a[4]; // "Access Denied"
    _m.style.color = "red";
}
```

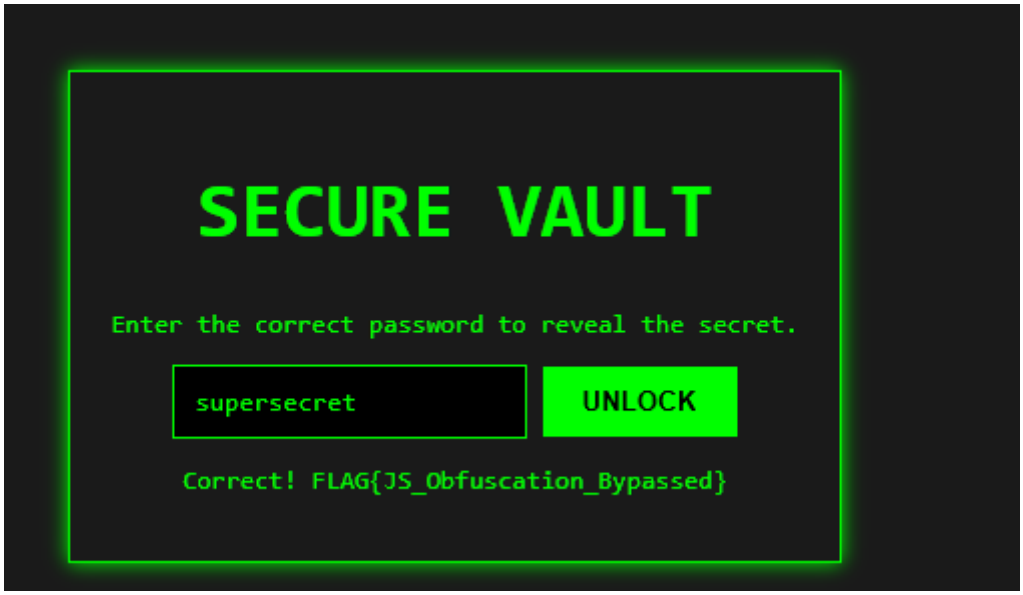
Y que el elemento 7 es la flag

```
<script> == $0
        // "Advanced" Client-Side Protection
        // The logic is obfuscated manually to simulate a typical "packer" or spaghetti code.

        var _0x4f2a = ["\x76\x61\x6C\x75\x65", "\x67\x65\x74\x45\x6C\x65\x6D\x65\x6E\x74\x42\x79\x49\x64", "\x70\x61\x73\x73", "\x6C\x65\x6E\x67\x74\x68", "\x41\x63\x63\x65\x73\x73\x20\x44\x65\x6E\x69\x65\x64", "\x6D\x73\x67",
"\x69\x6E\x6E\x65\x72\x54\x65\x78\x74", "\x46\x4C\x41\x47\x7B\x4A\x53\x5F\x4F\x62\x66\x75\x73\x63\x61\x74\x69\x6F\x6E\x5F\x42\x79\x70\x61\x73\x73\x65\x64\x7D", "\x43\x6F\x72\x72\x65\x63\x74\x21\x20",
"\x73\x75\x70\x65\x72\x73\x65\x63\x72\x65\x74"];

        // Decoded strings:
        // 0: value, 1: getElementById, 2: pass, 3: length, 4: Access Denied, 5: msg, 6: innerText
```

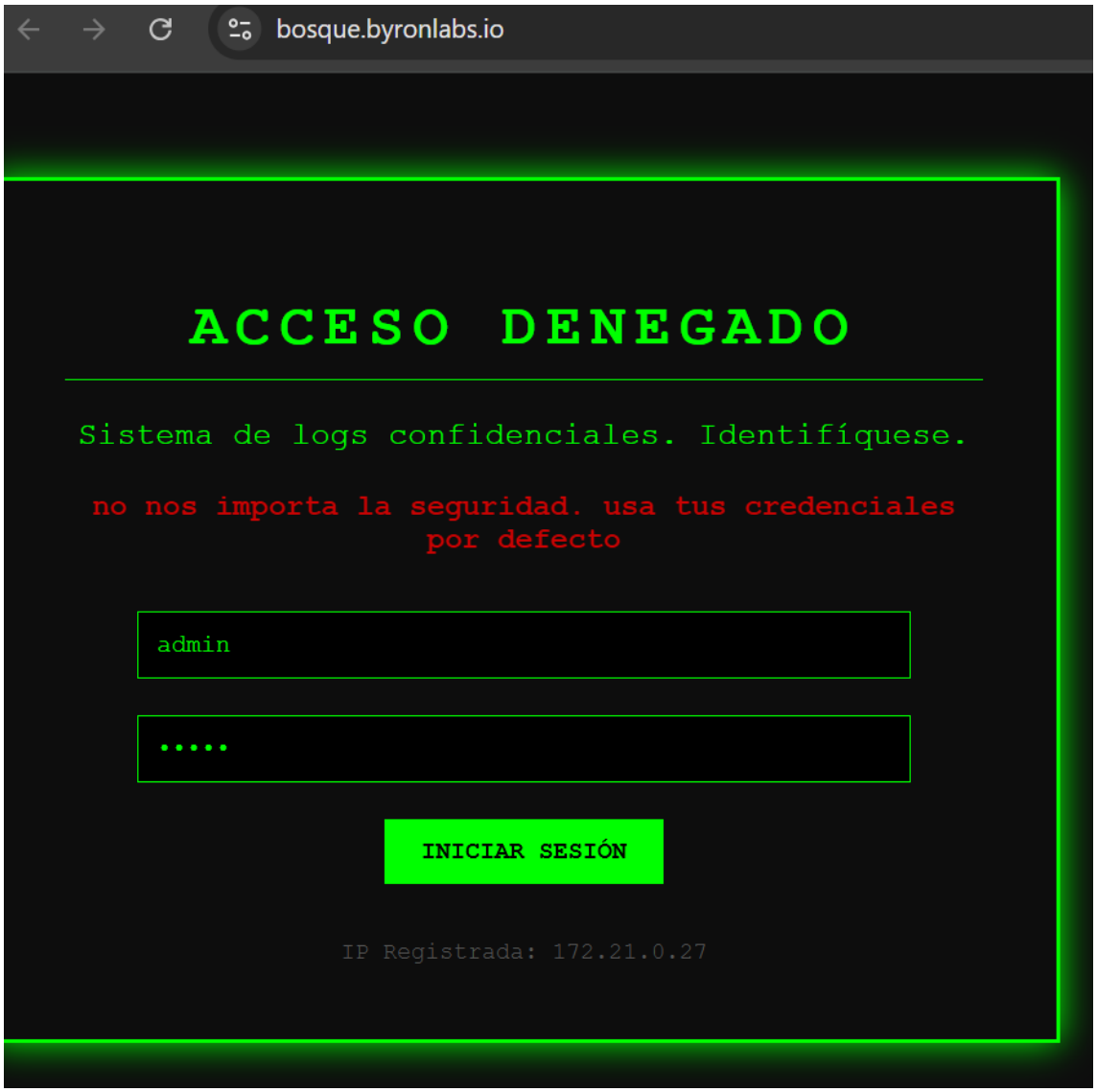
desofuscamos el array

```
var _0x4f2a = [
"value",
"getElementById",
"pass",
"length",
"Access Denied",
"msg",
"innerText",
"FLAG{JS_Obfuscation_Bypassed}",
"Correct! ",
"supersecret"
];
```

Ponemos la contraseña o diectamente la flag que es el elemento 7

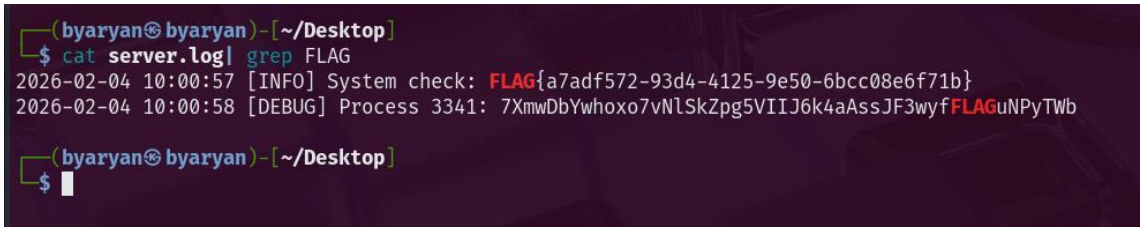


Reto 11

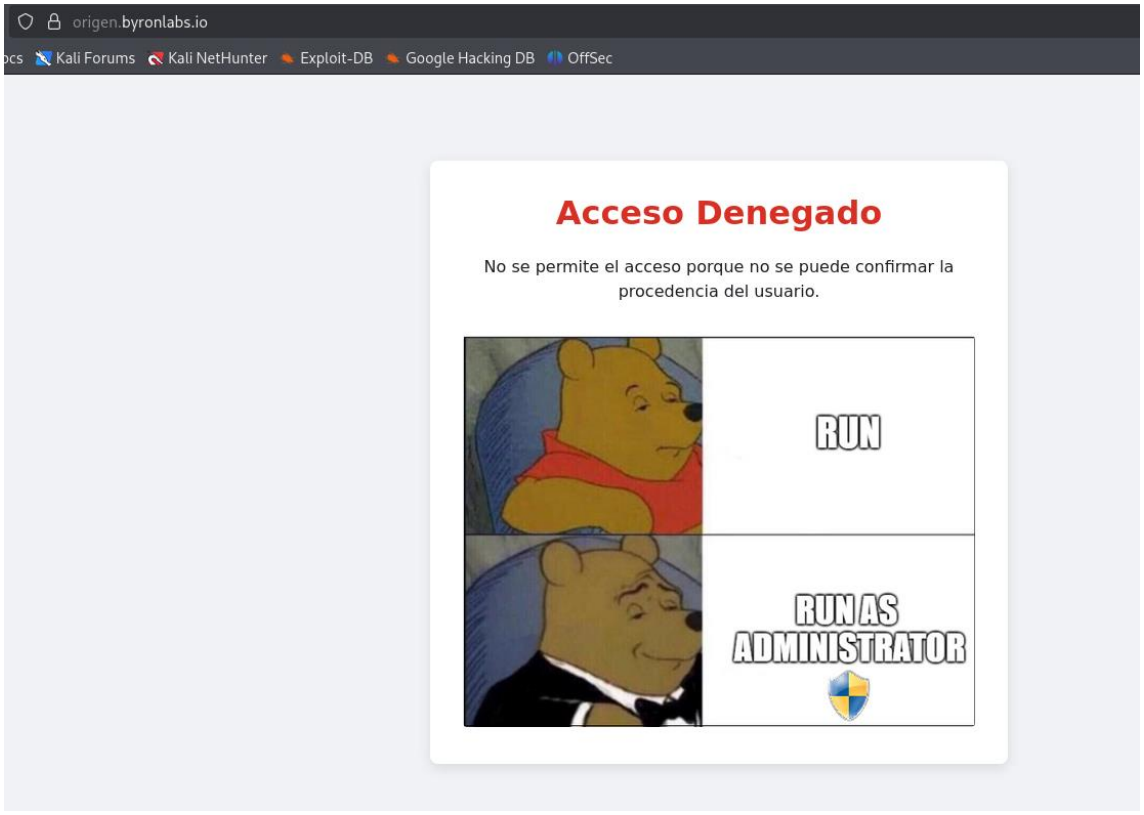


Probamos a poner la típica contraseña de admin/admin y se nos descarga un archivo .zip de logs.

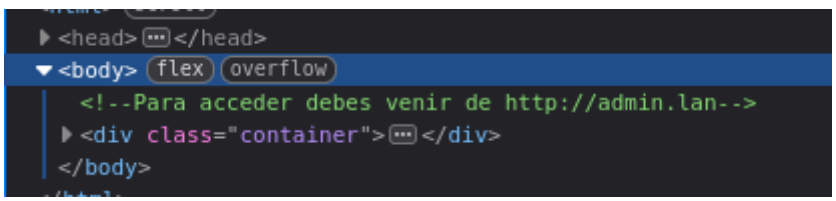
Hacemos un grep de este archivo buscando la flag



RETO12



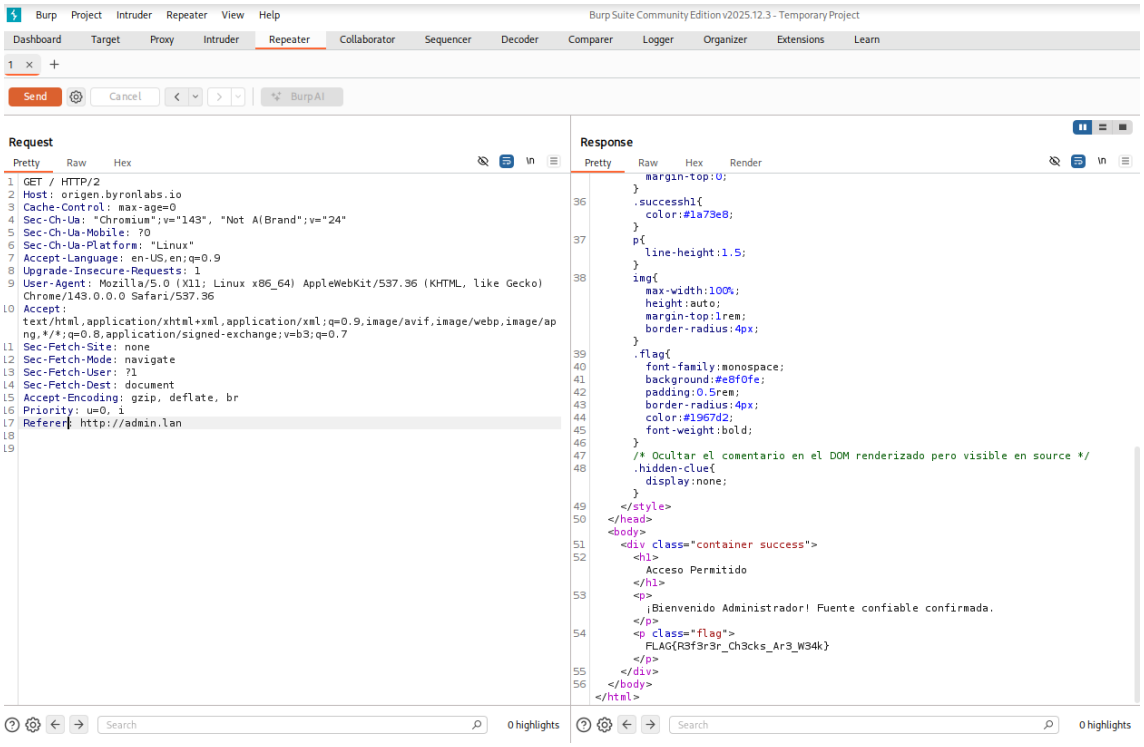
Inspeccionemos la pagina y obtenemos una pista



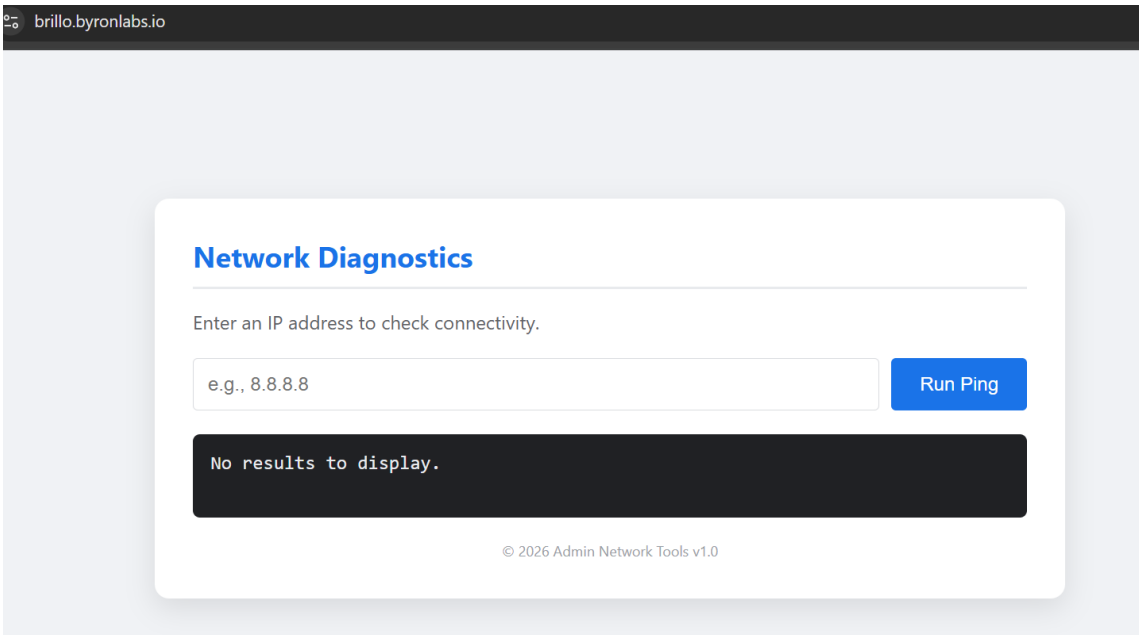
Así que probablemente tengamos que modificar la petición y cambiar el referer o el origin

Vamos a interceptar la petición con burpsuite

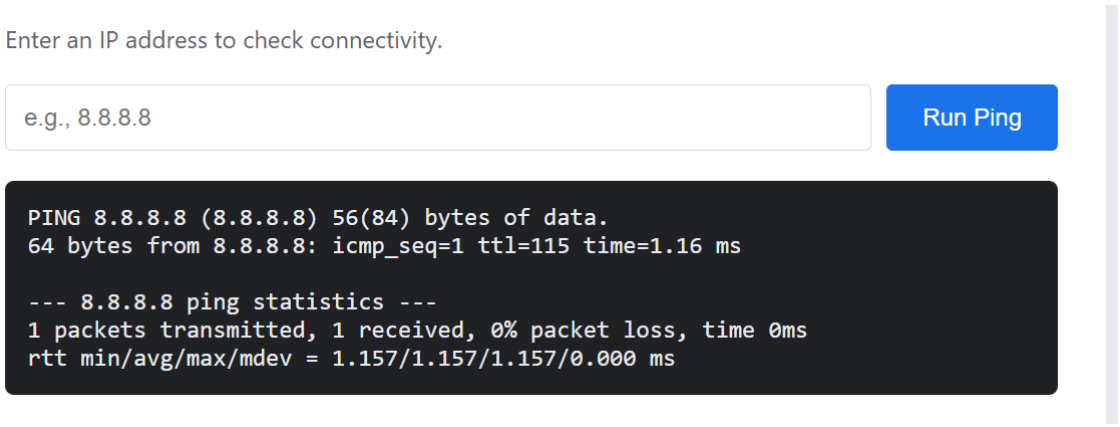
Probamos a enviar la cabecera Origin: http://admin.lan pero no es la correcta así que usamos la Referer: http://admin.lan y bingo, tenemos la flag



RETO 13



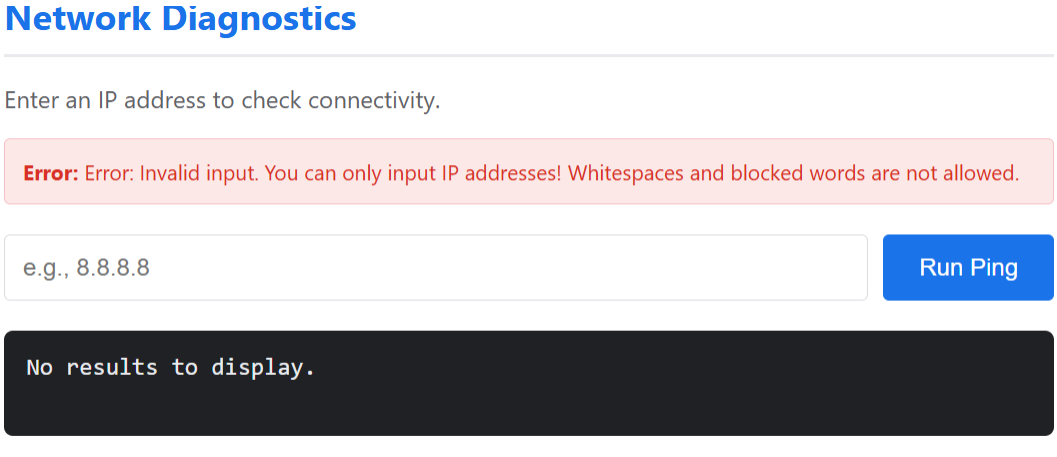
Inspeccionamos la pagina



Vemos la salida exacta del ping

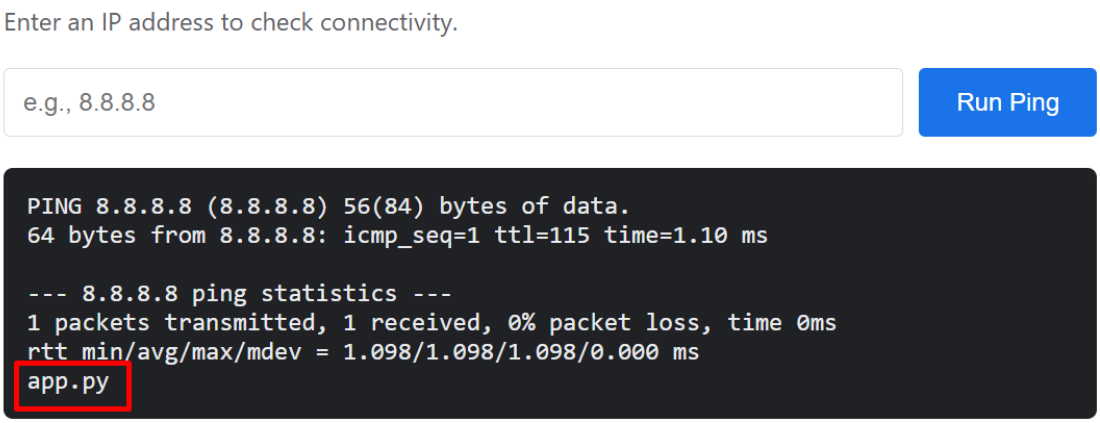
Así que vamos a probar a inyectar comandos por ejemplo ls

Comando usado 8.8.8.8; ls



Asique no podemos usar espacios en blanco

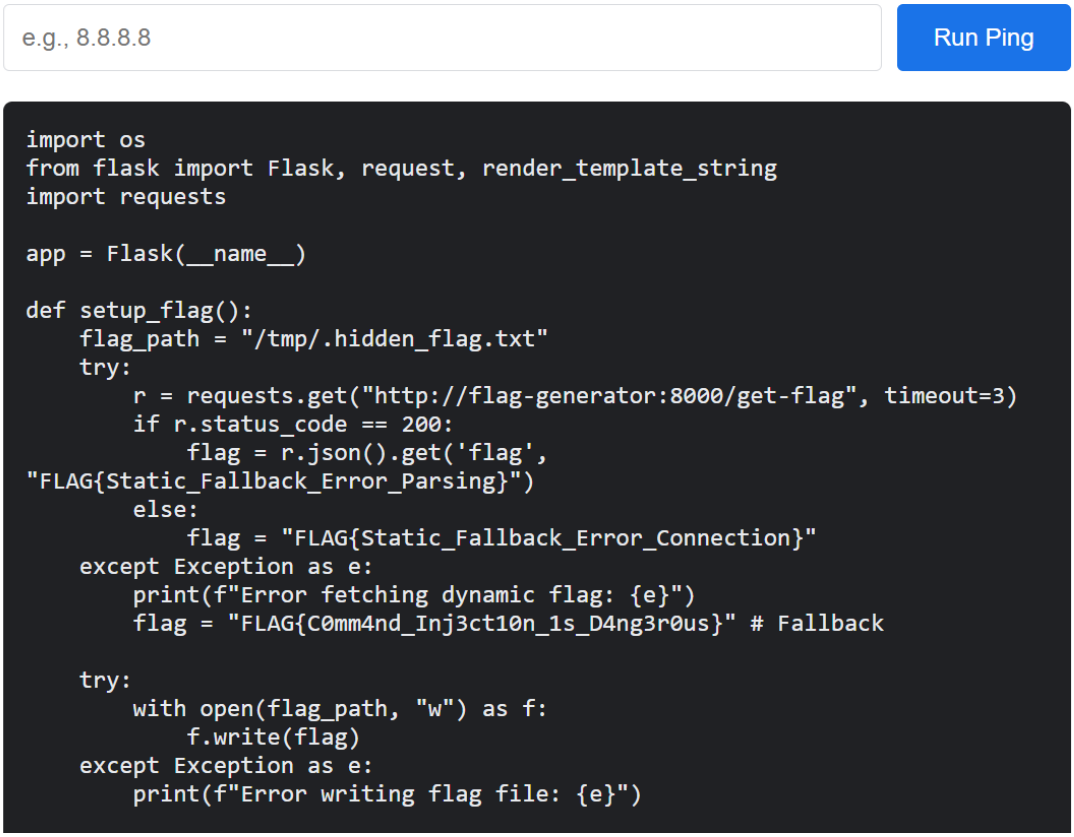
Comando usado 8.8.8.8;ls



Tenemos que imprimir ese archivo porque probablemente dentro este la flag y sin espacio

Para ello tenemos dos formas

Usar ${IFS} separador interno de Bash o cat<flag.txt si la redirección está permitida



Leyendo el código observamos varias flags y un path escondido que probablemente tenga nuestra flag, vamos a ir primero a por ello para eso lanzamos el siguiente comando

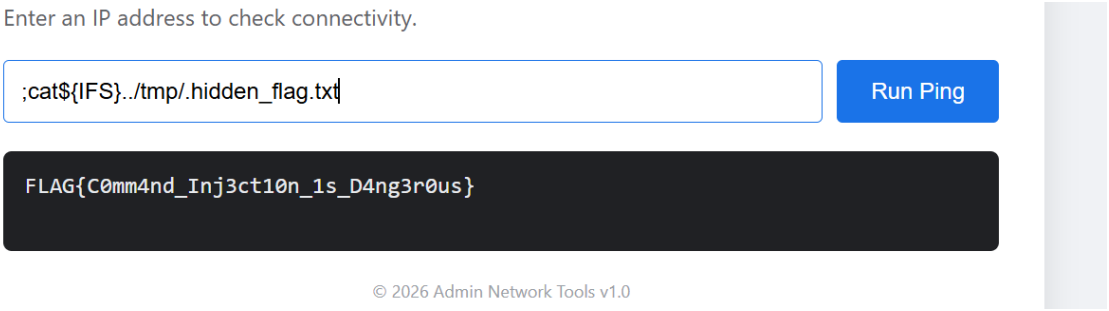
8.8.8.8;pwd para ver donde estamos

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=1.15 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.154/1.154/1.154/0.000 ms
/app
```

© 2026 Admin Network Tools v1.0

Probamos el comando

;cat${IFS}../tmp/.hidden_flag.txt



Forense

RETO 1

presunta comisión de un delito de malversación de caudales públicos y blanqueo de capitales, detectado en el marco de la Operación "Sombra". Las diligencias se iniciaron a raíz de una denuncia anónima recibida el pasado 15 de noviembre de 2025, donde se alertaba sobre irregularidades contables en la empresa pública "Urbanismo y Gestión S.L.". A continuación se exponen los hechos investigados, la identidad de los sujetos implicados y el análisis de la documentación financiera incautada durante el registro efectuado el día 3 de enero de 2026.

SUJETOS INVESTIGADOS

1. [redacted], con DNI [redacted]. En calidad de Director Financiero. Se le atribuye la autoría intelectual del desvío de fondos. 2. [redacted], con DNI [redacted]. En calidad de testaferro. Figura como administradora única de las sociedades pantalla utilizadas para el blanqueo. 3. [redacted]. Contable de la organización. Encargado de maquillar los libros de cuentas.

HECHOS INVESTIGADOS

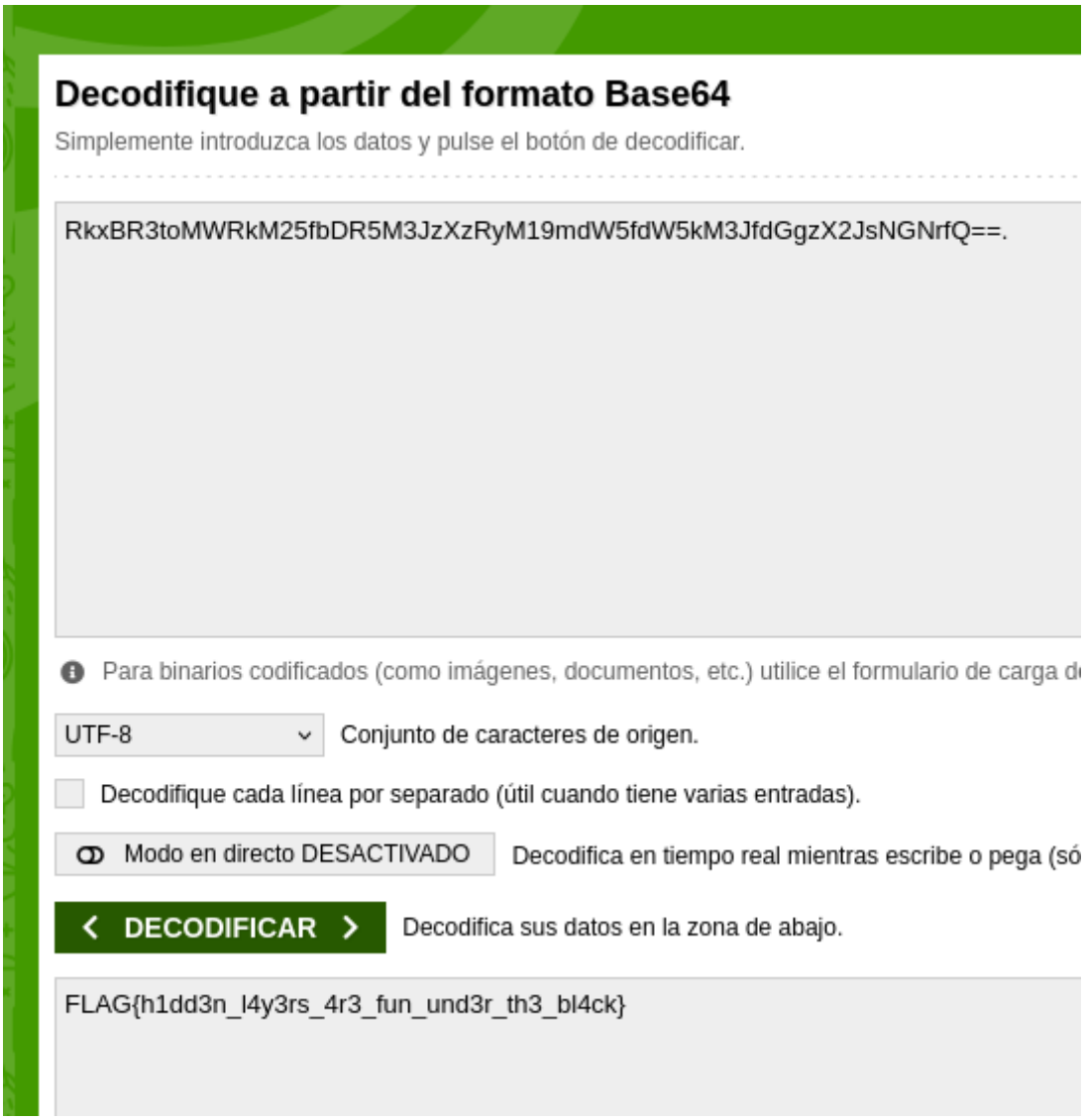
Durante el periodo comprendido entre 2023 y 2025, se han detectado transferencias periódicas desde las cuentas de la entidad pública hacia cuentas radicadas en paraísos fiscales. El modus operandi consistía en la facturación de servicios de consultoría inexistentes a través de la empresa instrumental "Consulting Global Future Ltd.". El importe total defraudado asciende a [redacted]. Los fondos eran posteriormente reintroducidos en el circuito legal mediante la compra de inmuebles de lujo en la Costa del Sol. Se ha constatado que el sujeto [redacted] mantenía reuniones frecuentes en el Hotel [redacted] con intermediarios financieros para orquestar estas operaciones.

ANÁLISIS FINANCIERO

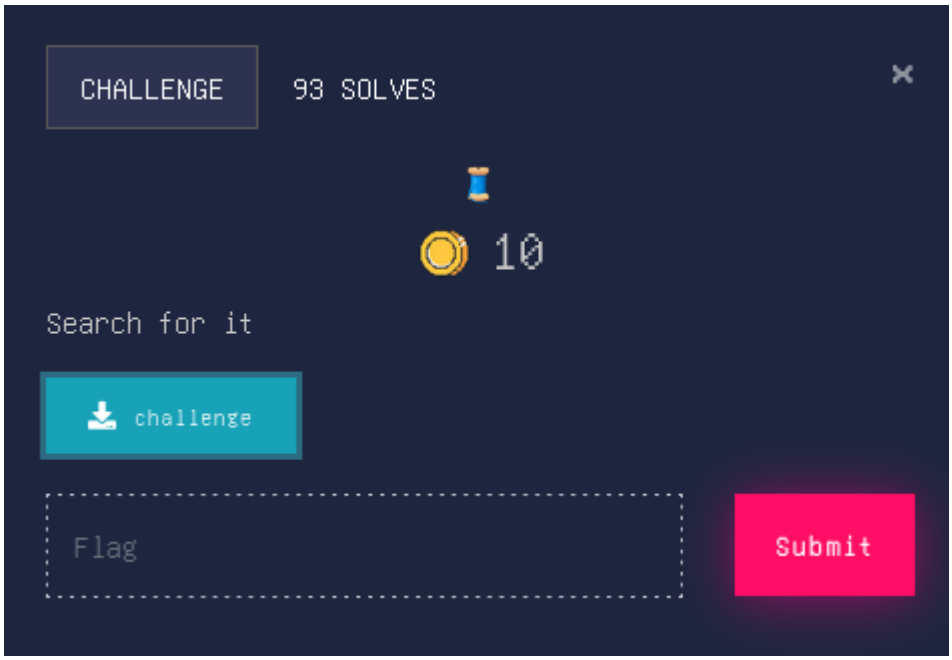
El análisis de los movimientos bancarios revela un patrón sistemático de fraccionamiento de capitales (smurfing) para eludir los controles de prevención del blanqueo de capitales. Se han identificado un total de 45 cuentas bancarias implicadas en la trama. Nota interna: La bandera de referencia para este caso es [redacted]. Utilizar para descifrado de archivos encriptados encontrados en el servidor principal. Las auditorías externas

Si copiamos y pegamos los bloques en negro podemos leer lo que hay detrás o incluso inspeccionando la pagina y encontramos algo sospechoso RkxBR3toMWRkM25fbDR5M3JzXzRyM19mdW5fdW5kM3JfdGgzX2JsNGNrfQ==

probablemente sea un flag codificada



RETO2




```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ exiftool challenge
ExifTool Version Number         : 13.44
File Name                       : challenge
Directory                       : .
File Size                       : 10 MB
File Modification Date/Time     : 2026:02:11 22:50:18+01:00
File Access Date/Time           : 2026:02:11 22:50:16+01:00
File Inode Change Date/Time     : 2026:02:11 22:50:18+01:00
File Permissions                : -rw-rw-r--
File Type                       : ELF executable
File Type Extension             :
MIME Type                       : application/octet-stream
CPU Architecture                : Unknown (168)
CPU Byte Order                  : Unknown (0)
Object File Type                : Unknown (6899)
CPU Type                        : Unknown (25834)

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ strings challenge | grep FLAG
FLAG{Y0u_V3ry_Cl0s3_N0w}
FLAG{Th1s_Is_4_F4k3_Fl4g_S0rry}
lFLAG{N0p3_N0t_Th1s_0n3_E1th3r}
FLAG{Str1ngs_C0mm4nd_1s_P0w3rful}
FLAG{Alm0st_Th3r3_But_N0t_Qu1t3}
FLAG{K33p_D1gg1ng_Y0u_Ar3_Cl0s3}
```

Miramos metadatos y strings del archivo y encontramos flags, las probamos para ver cual es aunque parece ser FLAG{Str1ngs_C0mm4nd_1s_P0w3rful}

RETO 3


```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ exiftool malicious_invoice.pdf
ExifTool Version Number         : 13.44
File Name                       : malicious_invoice.pdf
Directory                       : .
File Size                       : 961 bytes
File Modification Date/Time     : 2026:02:11 22:53:53+01:00
File Access Date/Time           : 2026:02:11 22:53:54+01:00
File Inode Change Date/Time     : 2026:02:11 22:53:53+01:00
File Permissions                : -rw-rw-r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.7
Linearized                      : No
Warning                         : Invalid xref table

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ strings  malicious_invoice.pdf | grep DLAG

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ strings  malicious_invoice.pdf | grep flag
flag.txts
flag.txtPK
```

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ binwalk malicious_invoice.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PDF document, version: "1.7"
683           0×2AB           Zip archive data, at least v2.0 to extract, compressed size: 41, uncompressed size: 39, name: flag.txt
762           0×2FA           Zip archive data, at least v2.0 to extract, compressed size: 29, uncompressed size: 29, name: loader.js
939           0×3AB           End of Zip archive, footer length: 22
```

Vemos que tiene un archivo flag.txt incrustado así que vamos a obtenerlo para eso usamos

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ dd if=malicious_invoice.pdf of=flag.zip bs=1 skip=683
278+0 records in
278+0 records out
278 bytes copied, 0.000725196 s, 383 kB/s

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ ls
byaryan.ovpn  challenge  evidencias  evidencias.zip  flag.txt  flag.zip

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ unzip flag.zip
Archive:  flag.zip
error [flag.zip]:  missing 683 bytes in zipfile
  (attempting to process anyway)
error: invalid zip file with overlapped components (possible zip bomb)
```

Como no nos deja descomprimir porque faltan 683 vamos a intentar descomprimir el pdf

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ unzip malicious_invoice.pdf
Archive:  malicious_invoice.pdf
replace flag.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: Y
  inflating: flag.txt
  inflating: loader.js
```

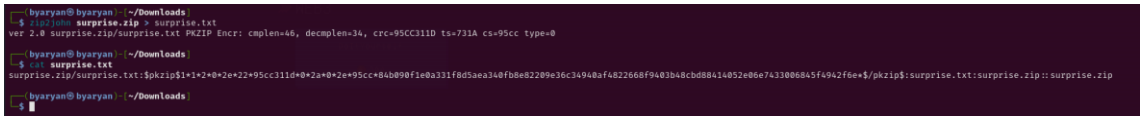
Y obtenemos nuestra flag

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ cat flag.txt
FLAG{P0lygl0t_F1l3s_Ar3_Tr1cky_Malw4r3}
```

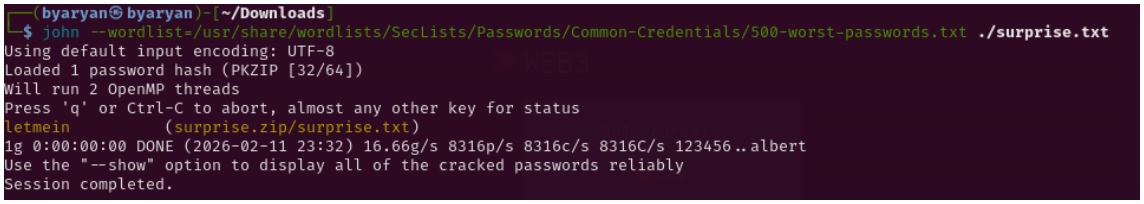
RETO4

Clonamos el repositorio así ya lo tenemos para próximas veces

Usamos zip2john para pasar el hash del zip a un txt y después crackearlo con john y la lista de seclist



Usamos el archivo de 500 peroes contrseñas y obtenemos la contrasñea del zip



Descomprimimos el zip una vez tenemos la contraseña y ya tendríamos la flag

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ unzip surprise.zip
Archive:  surprise.zip
[surprise.zip] surprise.txt password:
replace surprise.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: surprise.txt

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ cat surprise.txt
FLAG{wuh32dDdsSSDAihsdsi767whhdwo}
```

Reto 5


```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ exiftool image.img
ExifTool Version Number         : 13.44
File Name                       : image.img
Directory                       : .
File Size                       : 24 MB
File Modification Date/Time     : 2021:02:09 10:08:12+01:00
File Access Date/Time           : 2021:02:09 10:08:12+01:00
File Inode Change Date/Time     : 2026:02:11 23:37:29+01:00
File Permissions                : -rw-r--r--
Error                           : Unknown file type

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ strings image.img | grep FLAG
```

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ binwalk image.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
69632         0x11000         Zip archive data, at least v2.0 to extract, compressed size: 22182, uncompressed size: 23490, name: Secreto.png
91948         0x1672C         End of Zip archive, footer length: 22
92160         0x16800         JPEG image data, JFIF standard 1.01
104448        0x19800         JPEG image data, JFIF standard 1.01
112640        0x1B800         JPEG image data, JFIF standard 1.01
```

Vamos a extraer el secreto.png del image.img

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ dd if=image.img of=secreto.zip bs=1 skip=0 count=91970
91970+0 records in
91970+0 records out
91970 bytes (92 kB, 90 KiB) copied, 0.231578 s, 397 kB/s

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ unzip secreto.zip
Archive:  secreto.zip
warning [secreto.zip]:  69632 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  inflating: Secreto.png
```

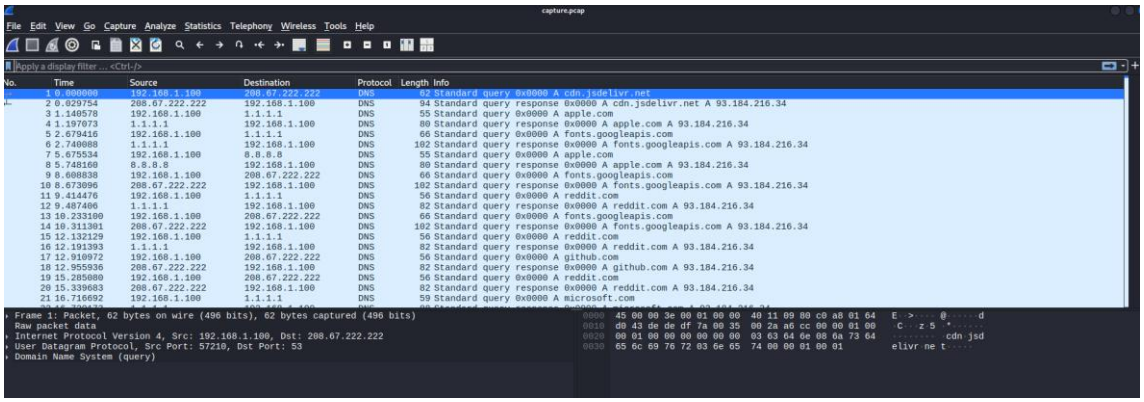
```
1 Se ha detectado una nueva cepa del virus que va a tranformar a todos los infectados en zombies.
2 Debemos de ponernos los que podamos a salvo lo antes posible
3 Empieza por los conocidos amigos y familia con la clave flag{IdontForget$$}
4 Luego personas que nos puedan ser utiles para poder sobrevivir
5 Esto es de extrema urgencia y alta confidencialidad..
6
7 TOP SECRET!!
```

obtenemos la flag

Flag{IdontForget$$}

RETO6

Captura de tráfico wireshark



Y vemos que hay peiciones que tienen un data-suspicious.local

Agrupando obtenemos que son:
RkxBR3tE.exfil-data.suspicious.local

TlNfYzB2.exfil-data.suspicious.local

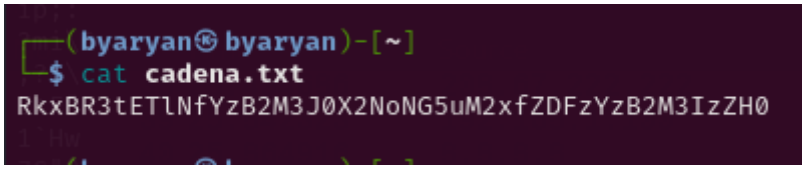
M3J0X2No.exfil-data.suspicious.local

NG5uM2xf.exfil-data.suspicious.local

ZDFzYzB2.exfil-data.suspicious.local

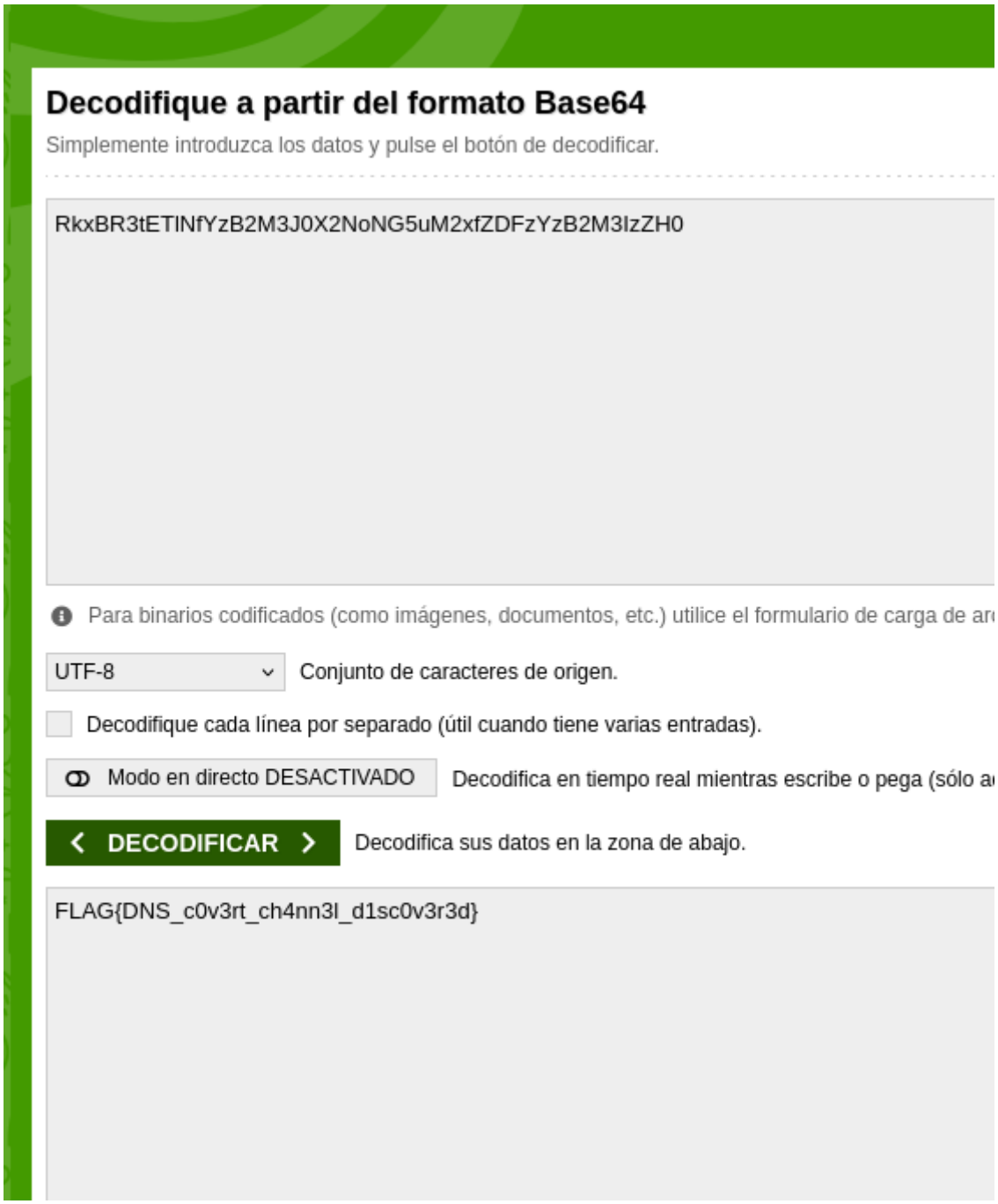
M3IzZH0.exfil-data.suspicious.local

Si nos quedamos con la primera parte parece que es la flag codificada



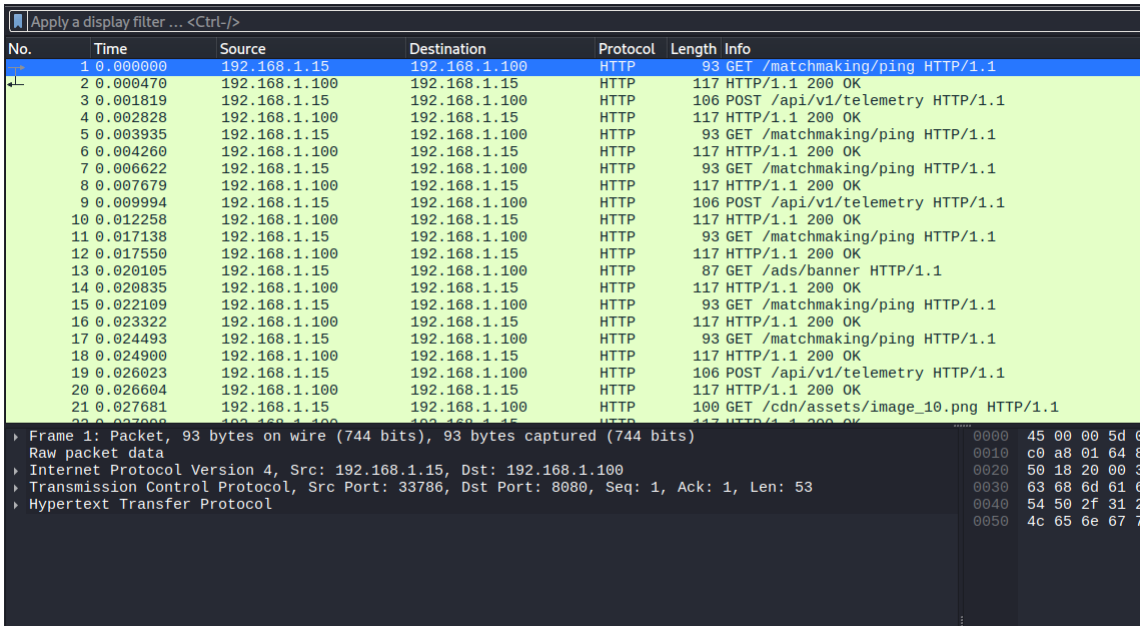
Vamos a probar

Y efectivamente

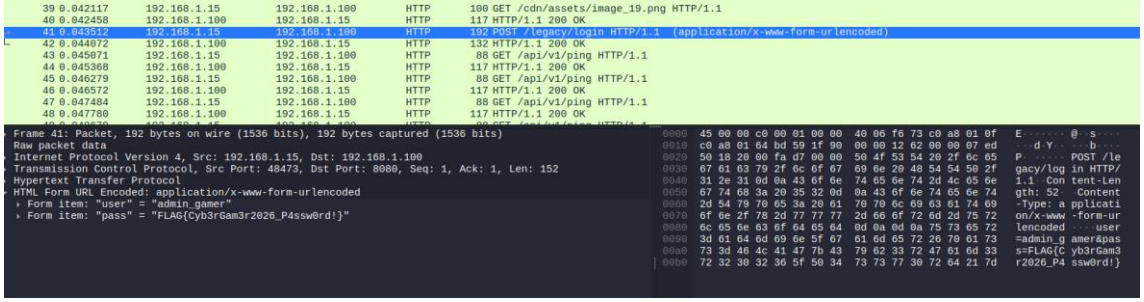


RETO7

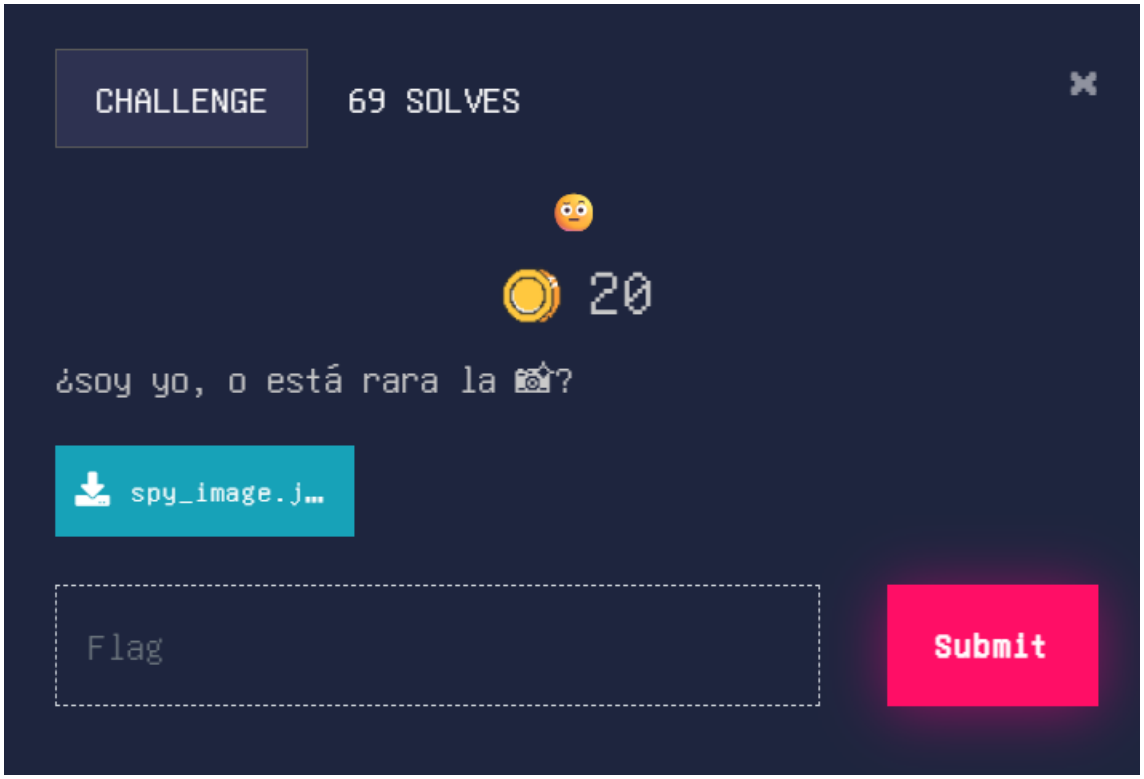
Descargamos la nueva captura de Wireshark



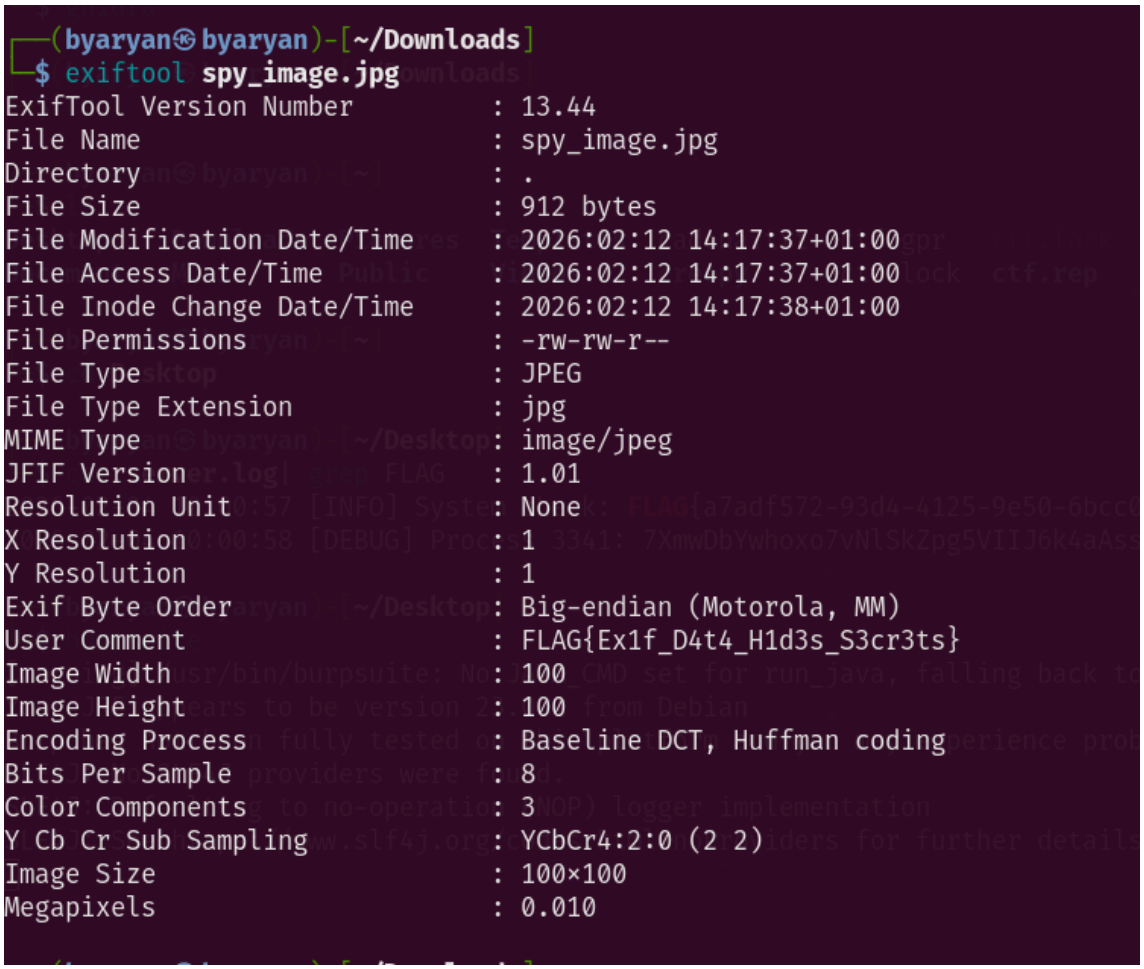
Y revisando las peticiones observamos en claro la flag en la password



RETO 9

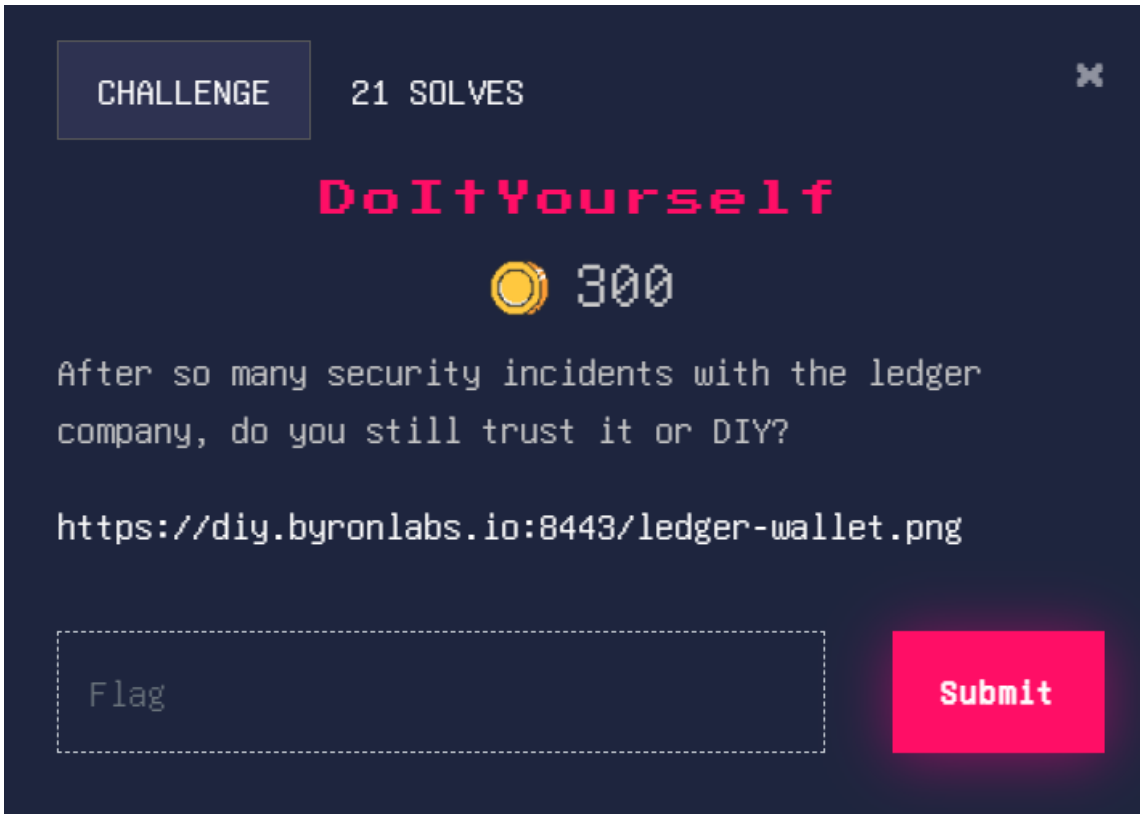


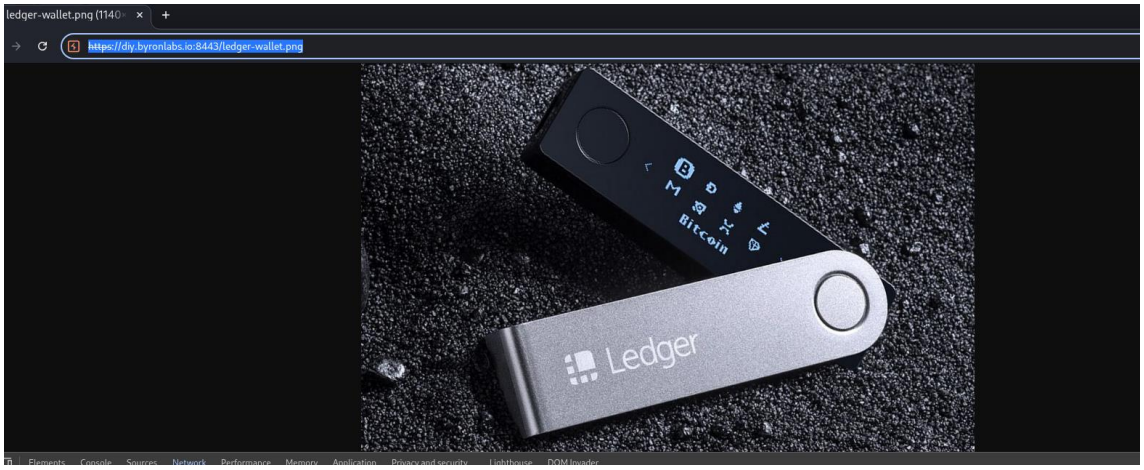
Miramos metadatos


(byaryan㉿byaryan)-[~/Downloads]
$ exiftool spy_image.jpg
ExifTool Version Number : 13.44
File Name : spy_image.jpg
Directory : .
File Size : 912 bytes
File Modification Date/Time : 2026:02:12 14:17:37+01:00
File Access Date/Time : 2026:02:12 14:17:37+01:00
File Inode Change Date/Time : 2026:02:12 14:17:38+01:00
File Permissions : -rw-rw-r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Resolution Unit : None
X Resolution : 1
Y Resolution : 1
Exif Byte Order : Big-endian (Motorola, MM)
User Comment : FLAG{Ex1f_D4t4_H1d3s_S3cr3ts}
Image Width : 100
Image Height : 100
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 100×100
Megapixels : 0.010

WEB3

Reto





Nos descargamos la imagen con wget

Observamos metadatos de l a imgen

```
┌──(byaryan㉿byaryan)-[~]
└─$ exiftool ledger-wallet.png
ExifTool Version Number         : 13.44
File Name                       : ledger-wallet.png
Directory                       : .
File Size                       : 1385 kB
File Modification Date/Time     : 2026:02:11 16:17:11+01:00
File Access Date/Time           : 2026:02:11 21:56:49+01:00
File Inode Change Date/Time     : 2026:02:11 21:56:49+01:00
File Permissions                : -rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1140
Image Height                    : 720
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Image Size                      : 1140×720
Megapixels                      : 0.821
```

Miramos los strings de la imagen por si hay alguno relacionado con la FLAG

```
┌──(byaryan㉿byaryan)-[~]
└─$ strings ledger-wallet.png | grep FLAG
```

pero nada

por ultimo

Vemos si hay datos en la imagen

```
┌──(byaryan㉿byaryan)-[~]
└─$ binwalk ledger-wallet.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1140 x 720, 8-bit/color RGBA, non-interlaced
41            0x29            Zlib compressed data, default compression
313402        0x4C83A         JBOOT STAG header, image id: 8, timestamp 0x60C18E, image size: 2366650059 bytes, image JBOOT checksum: 0x188C, header JBOOT checksum: 0x368
```

Y observamos que hay un JBOOT STAG header que se refiere a una estructura de datos específica utilizada en el cargador de arranque

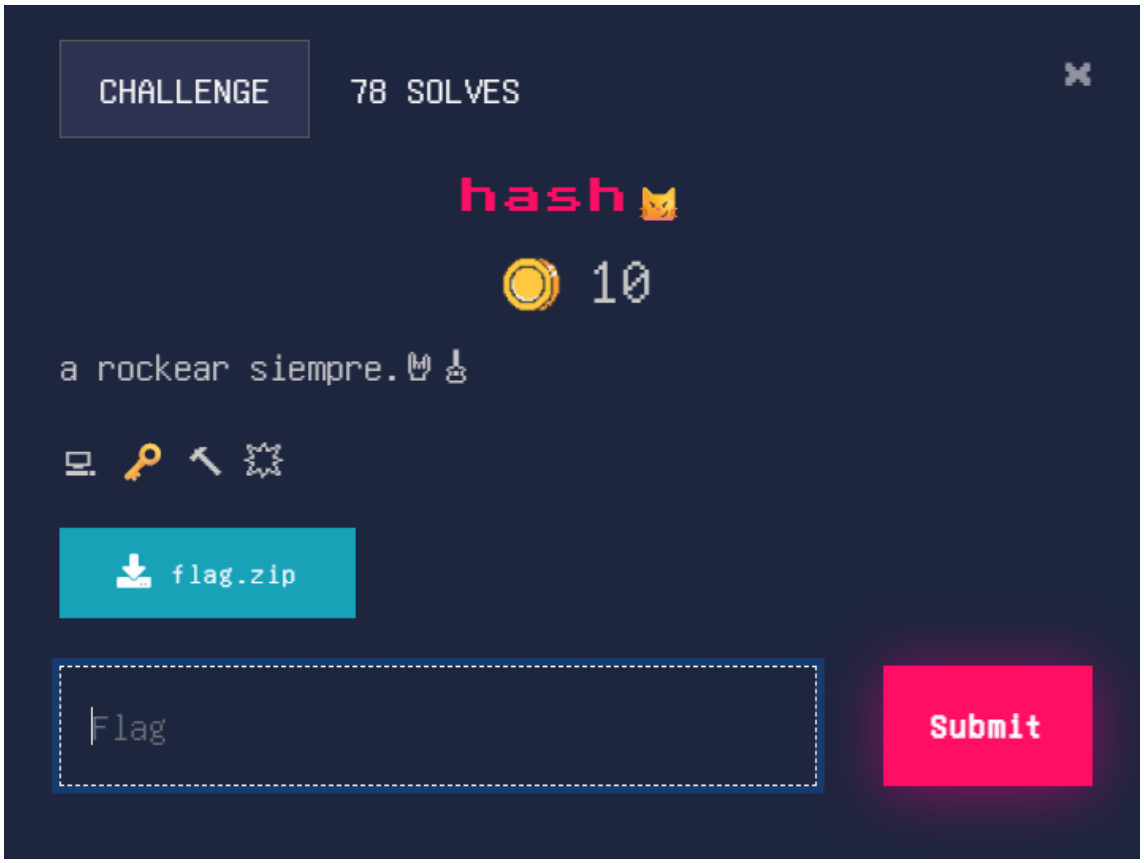
Usamos dd para exraer los datos

```
┌──(byaryan㉿byaryan)-[~]
└─$ sudo dd if=ledger-wallet.png of=jboot.bin bs=1 skip=313402
1071427+0 records in
1071427+0 records out
1071427 bytes (1.1 MB, 1.0 MiB) copied, 2.01759 s, 531 kB/s
```

Una vez tenemos el binario hacemos lo mismo y al no enconrar nada vemos a ver si en ghidra obtenemos algo

CRIPTOGRAFIA

Reto 1



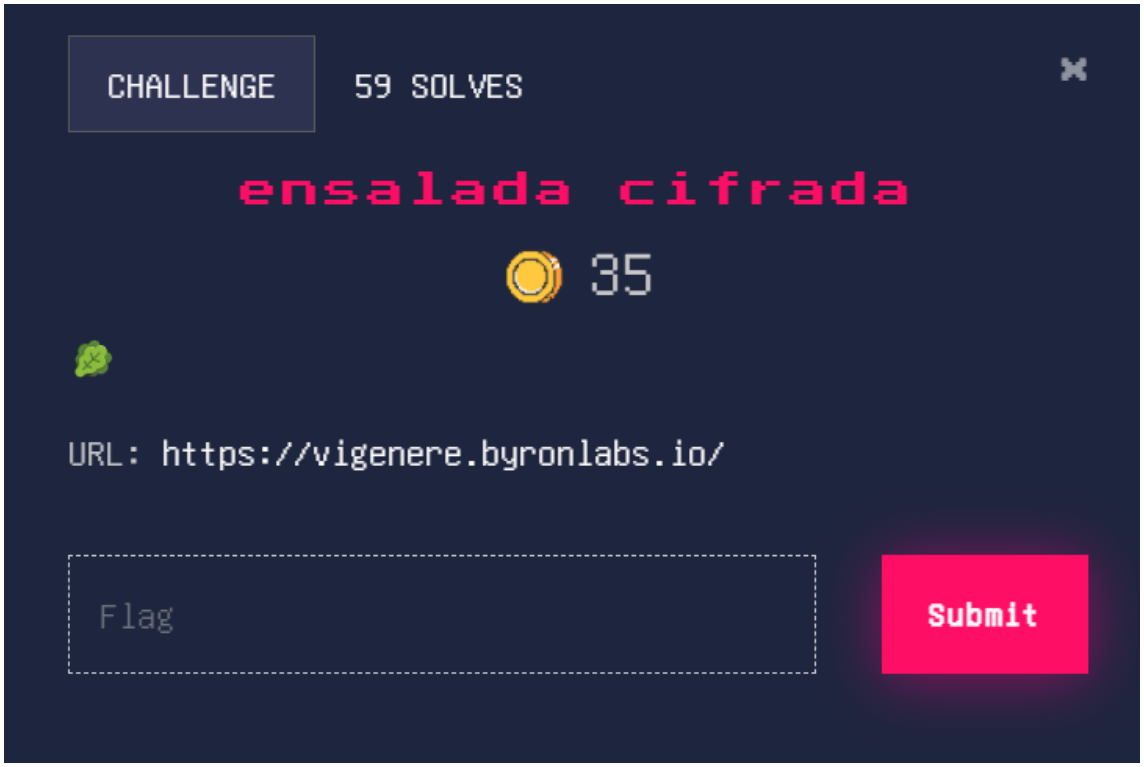
Con zip2john Extraemos el hash del .zip con contraseña y lo guardamos en otro archivo

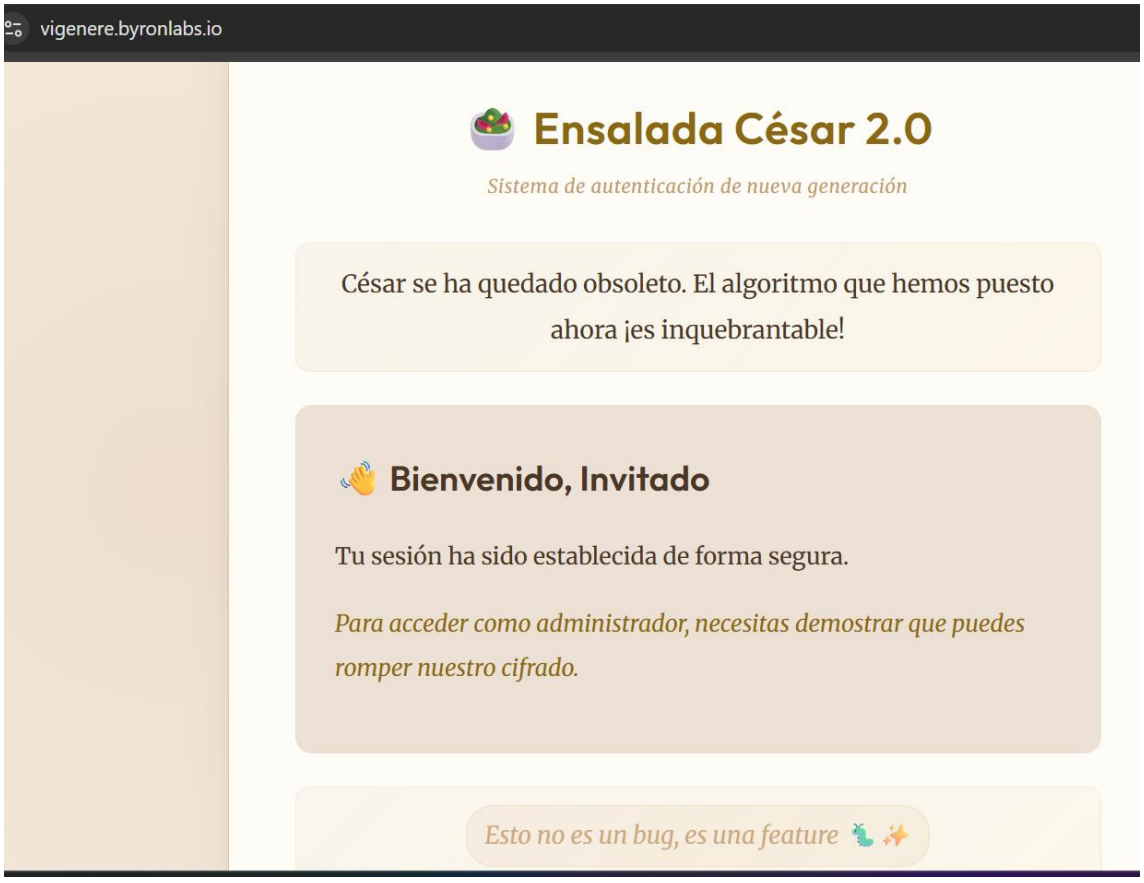
Y después usamos john junto con la wordlist rockyou para crackear el hash

```
┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ 7z x flag.zip

7-Zip 25.01 (x64) : Copyright (c) 1999-2025 Igor Pavlov : 2025-08-03
 64-bit locale=C.UTF-8 Threads:2 OPEN_MAX:1024, ASM

Scanning the drive for archives:
1 file, 196 bytes (1 KiB)

Extracting archive: flag.zip
--
Path = flag.zip
Type = zip
Physical Size = 196

Would you like to replace the existing file:
  Path:     ./flag.txt
  Size:     193 bytes (1 KiB)
  Modified: 2026-02-11 18:17:28
with the file from archive:
  Path:     flag.txt
  Size:     32 bytes (1 KiB)
  Modified: 2026-02-09 17:31:32
? (Y)es / (N)o / (A)lways / (S)kip all / A(u)to rename all / (Q)uit? Y


Enter password (will not be echoed):
Everything is Ok

Size:       32
Compressed: 196

┌──(byaryan㉿byaryan)-[~/Downloads]
└─$ cat flag.txt
FLAG{Z1p_Cr4ck1ng_1s_E4sy_P34sy}
```

RETO 2

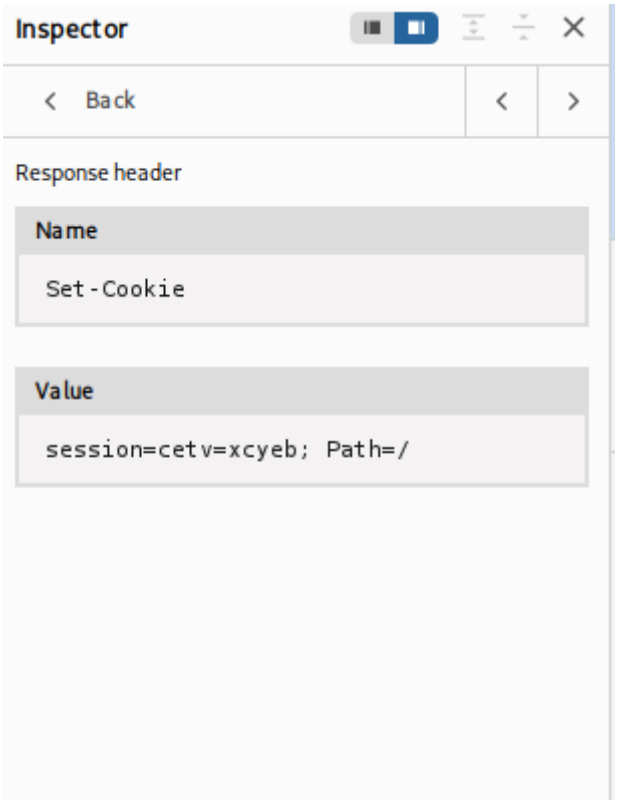
RETO 3





Inspeccionamos la página y a simple vista no observamos nada

Analizando la petición desde Burpsuite vemos que nos devuelven una cabecera de sesión un tanto sospechosa



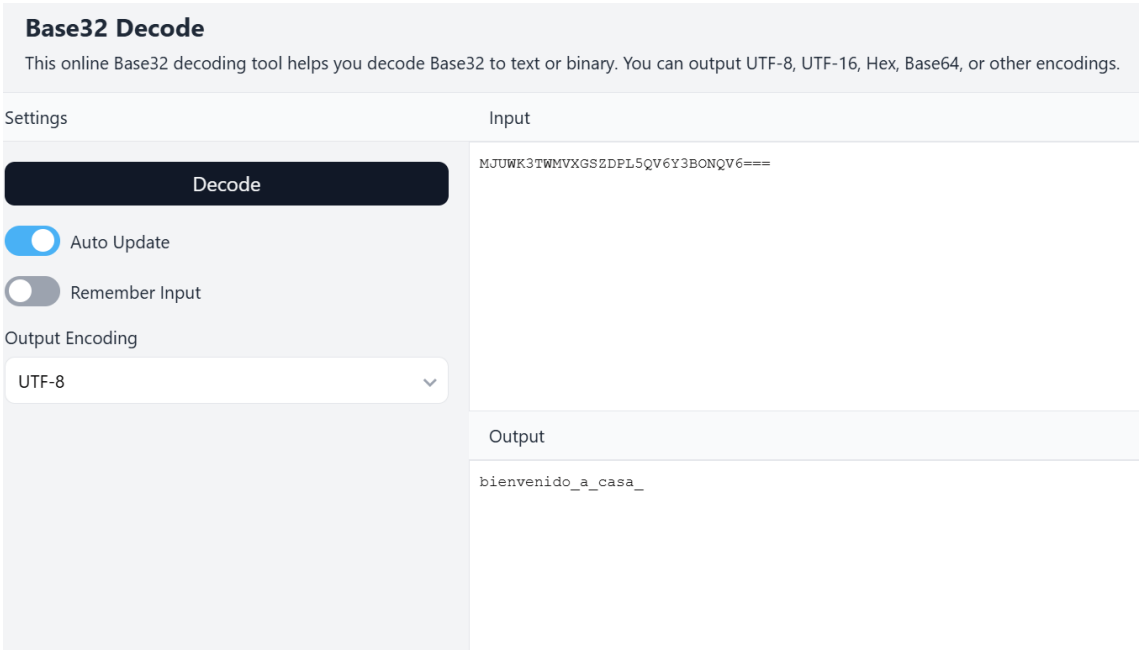
Cetv=xcyeb

Que puede ser un estilo

role=

user=

reto



Reversing

RETO1

Usamos el comando file para ver si el binario esta stripped y si será fácil hacer un reversing

Y efectivamente vemos que esta stripped

Vamos a ver los strings en el binario

Y observamos que parece que hay strings en base64

cjN2M3JzM191YWhfaWZfeTB1X2M0bg==

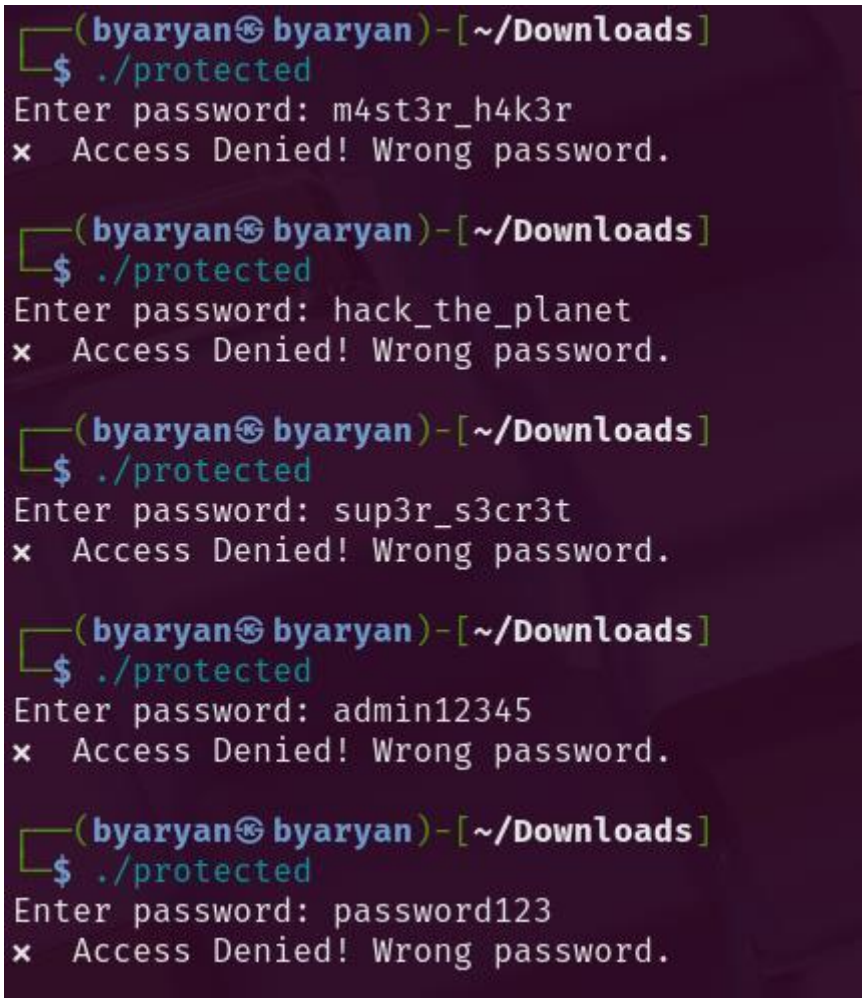
bTRzdDNyX2g0azNy -> m4st3r_h4k3r

aGFja190aGVfcGxhbmV0 -> hack_the_planet

c3VwM3JfczNjcjN0 -> sup3r_s3cr3t

YWRtaW4xMjM0NQ== -> admin12345

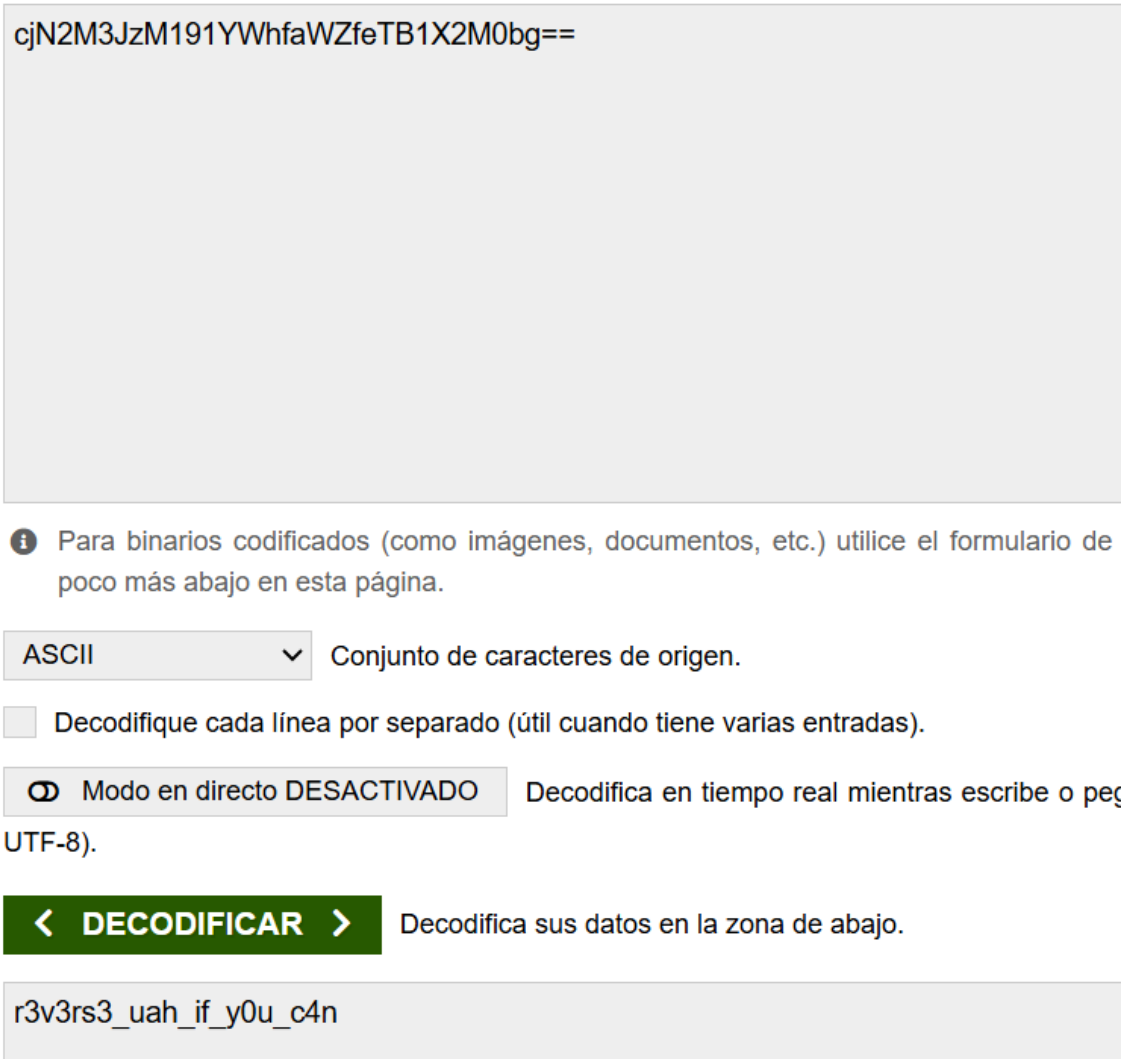
cGFzc3dvcmQxMjM= -> password123



Sabemos cuales la FLAG porque el primer string es lo que hay en el interior de la FLAG podríamos montarlo ya que ninguna de las contraseñas ha sido la correcta

Y esta seria la FLAG{r3v3rs3_uah_if_y0u_c4n}

El primer valor parece parte de la flag



Tras codificar todos los strings vemos que tenemos varias posibles contraseñas a utilizar así que probamos