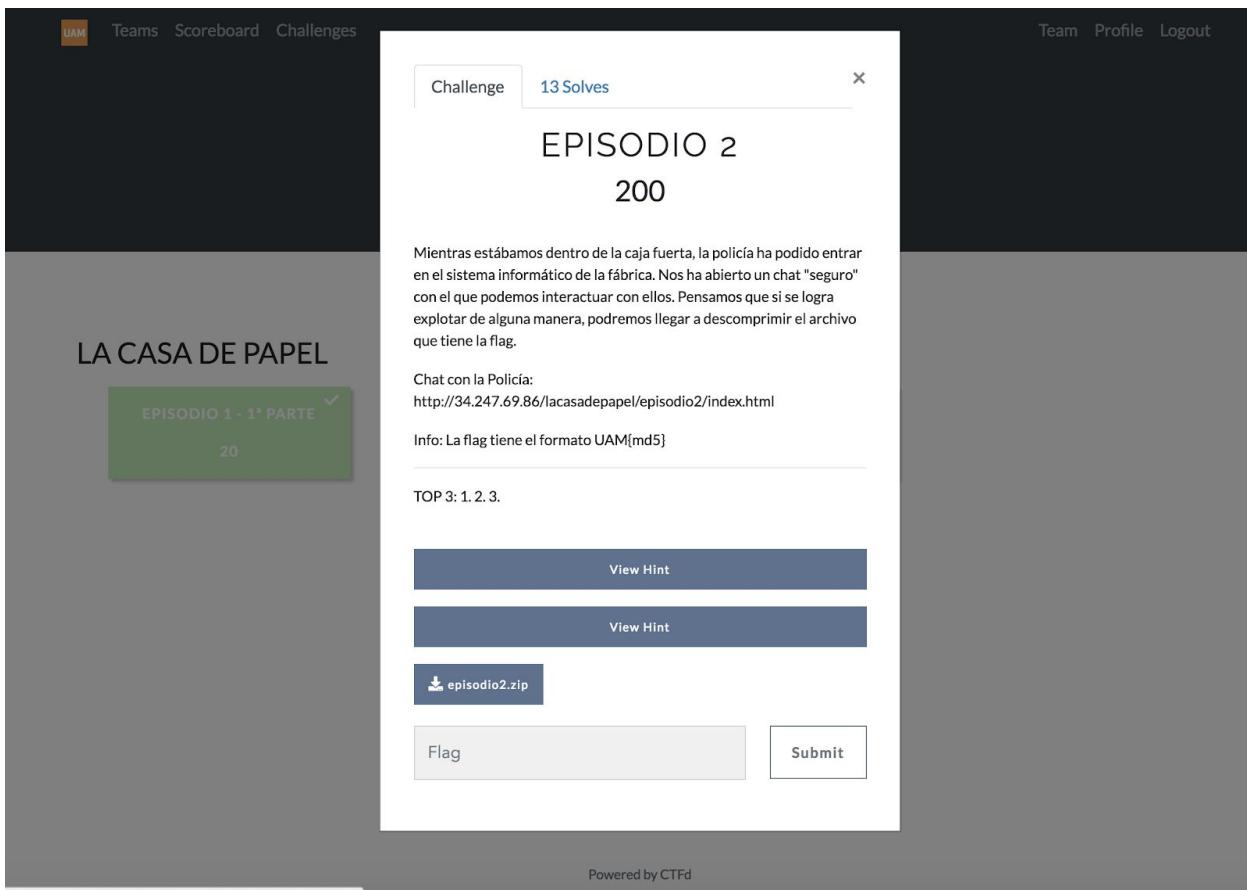


Episodio 2

Chat con la policía

Introducción	2
Interactuando con el chat de la policía.	2
Primera obtención de la clave de cifrado del Zip	3
Fichero de audio dentro del Zip	11
Vuelta al inicio. Segunda manera de obtener la contraseña del zip	19
Con las coordenadas, hasta la resolución.	22



Introducción

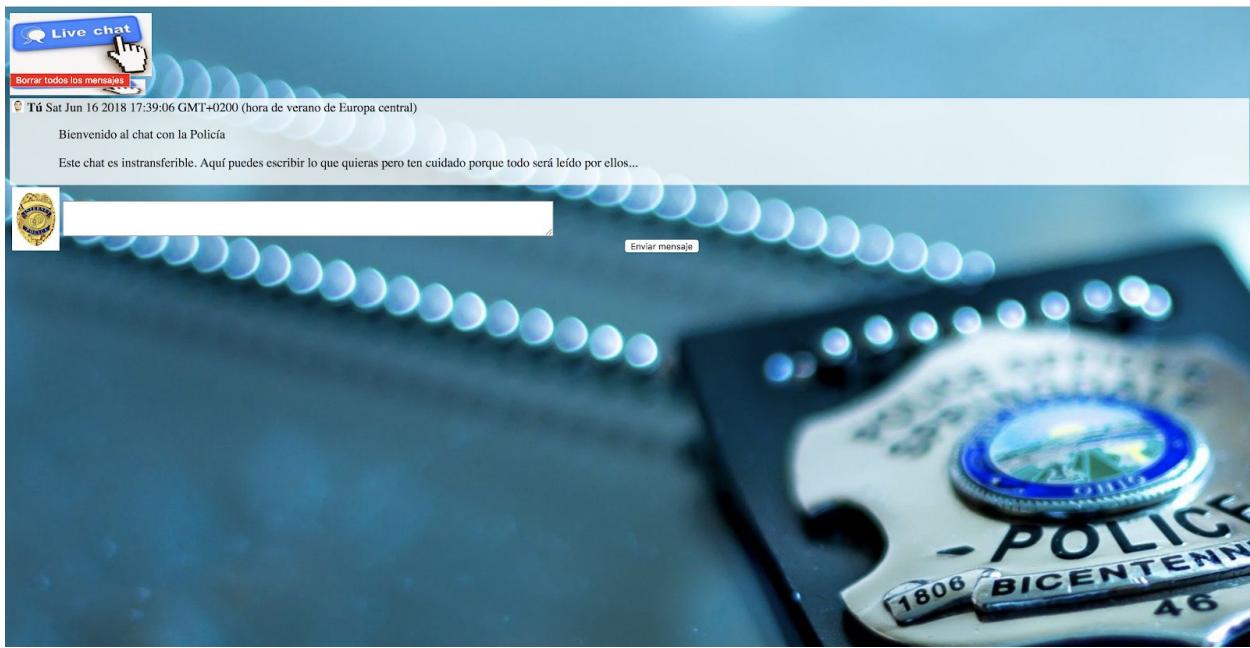
El reto que se nos presenta esta vez está relacionado con la comunicación que ha establecido la policía con nosotros, facilitandonos un chat en el que interactuar con ellos. Las indicaciones que nos dan nos informan de que es posible explotar alguna vulnerabilidad en el mismo, veamos qué podemos hacer al respecto

Interactuando con el chat de la policía.

Una vez que establecemos contacto con el chat de la policía en la URL

<http://34.247.69.86/lacasadepapel/episodio2/index.html>

se nos presenta un chat con el siguiente aspecto:



Sin duda es un trabajo de la administración, o de algún organismo público, ya que el diseño gráfico es espartano, con un cierto aire Retro de los 80, incluso con algún defecto en la visualización del ícono superior. No hay equivocación posible, estamos en la página correcta.

Primera obtención de la clave de cifrado del Zip

Empezamos con un análisis del contenido de dicho chat. Páginas referenciadas, código javascript e imágenes que aparecen son nuestros primeros objetivos.

El código HTML de la página no parece tener ningún tipo de comentario o texto oculto con pistas acerca del objetivo a alcanzar. Algunas funciones de JavaScript definidas en la propia página que parece se utilizan para mostrar y guardar los mensajes que enviamos al chat y para borrar dichos mensajes.

```

<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <script src="game-frame.js"></script>
    <link rel="stylesheet" href="styles.css">
    <script src="post-store.js"></script>
  </head>
  <body>
    <div id="post-container"></div>
    <form id="post-form">
      <input type="text" id="post-content" placeholder="Escribe tu mensaje...">
      <input type="button" value="Publicar" />
    </form>
    <div id="clear-form">Borrar todo</div>
  </body>
</html>

```

El objeto PostDB es el primero que nos llama la atención ya que el sufijo DB sugiere algún tipo de almacenamiento en base de datos y la vulnerabilidad requerida podría tratarse de un tipo de SQL-Injection o un ataque a la base de datos que se encuentre detrás del mantenimiento de los mensajes, por lo que investigamos un poco más dicho código.

La declaración de ese objeto en JS debe encontrarse en alguno de los ficheros que como ficheros de script están definidos en la sección cabecera (head) de la página.

```

<head>

  <meta charset="UTF-8">

  <script src="game-frame.js"></script>

  <link rel="stylesheet" href="styles.css">

  <script src="post-store.js"></script>

```

El más sugerente es el que se llama post-store así que continuamos la investigación sobre dicho fichero.

```
function Post(message) {
    this.message = message;
    this.date = (new Date()).getTime();
}

function PostDB(defaultMessage) {

    this._defaultMessage = defaultMessage || "";

    this.setup = function() {
        var defaultPost = new Post(defaultMessage);
        window.localStorage["postDB"] = JSON.stringify({
            "posts" : [defaultPost]
        });
    }

    this.save = function(message, callback) {
        var newPost = new Post(message);
        var allPosts = this.getPosts();
        allPosts.push(newPost);
        window.localStorage["postDB"] = JSON.stringify({
            "posts" : allPosts
        });

        callback();
        return false;
    }

    this.clear = function(callback) {
        this.setup();

        callback();
        return false;
    }

    this.getPosts = function() {
```

```
        return JSON.parse(window.localStorage["postDB"]).posts;
    }

    if(!window.localStorage["postDB"]) {
        this.setup();
    }
}
```

El código en si no nos aporta nada en especial. Nos indica que el sistema de almacenamiento que está utilizando para los mensajes es el Local Storage (`window.localStorage`) del navegador con lo que descartamos la hipótesis inicial de un ataque vía SQLi ya que sobre dicho sistema de almacenamiento local del navegador “no se pueden realizar”. Como testear la seguridad de este tipo de almacenamiento se puede leer en

[https://www.owasp.org/index.php/Test_Local_Storage_\(OTG-CLIENT-012\)](https://www.owasp.org/index.php/Test_Local_Storage_(OTG-CLIENT-012))

y algunos datos adicionales del mismo en el security cheat sheet de html5

https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet#Local_Storage

Comprobamos que los mensajes se almacenan en este sitio a través de las herramientas de desarrollo de Chrome, desde la pestaña Application y seleccionando en el menú de la derecha , bajo Storage, la opción de Local Storage.

The screenshot shows the Chrome DevTools Application tab open. On the left, there's a sidebar with sections for Application (Manifest, Service Workers, Clear storage), Storage (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), Cache (Cache Storage, Application Cache), and Frames (top). The main area displays a table of stored posts:

Key	Value
posts: [{}]	{message: "Prueba de mensaje-", date: 1529317735199, ...}
posts: [{}]	{message: "Prueba de mensaje-", date: 1529317735199, ...}
0: {...}	date: 1529163546773 message: "Bienvenido al chat con la Policía Este chat es intransferible. Aquí podrás interactuar con el personal de la fuerza de seguridad para obtener información sobre tu situación y recibir asistencia en caso de emergencia. ¡Tenemos tu seguridad en alta consideración!" date: 1529317735199 message: "Prueba de mensaje-"
1: {...}	date: 1529317735199 message: "Prueba de mensaje-"
2: {...}	date: 1529317745771 message: "Otra prueba mas"

Below the table, the DevTools interface includes tabs for Console, Rendering, Request blocking, Quick source, and What's New, with the Console tab currently selected. The bottom section shows the developer console with the following message:

```
1 message
  No user messages
  No errors
  No warnings
```

Como podemos ver los mensajes almacenados no tienen nada de especial, un formato JSON con una fecha y el texto que hemos introducido nosotros a través del chat.

Echamos un vistazo al otro código Javascript que vemos en la definición de la cabecera HTML , game-frame.js

The screenshot shows the browser developer console with the URL 34.247.69.86/lacasadepapel/episodio2/game-frame.js. The code is heavily obfuscated, appearing as a long string of characters. It contains several comments (//) and some recognizable words like 'function', 'var', and 'return'. The code is likely a compressed version of the original JavaScript file, possibly generated by a minifier or obfuscator.

¡Este código si que es interesante! Solo aparecen en él un cierto número de caracteres, para estar seguros de cuáles son esos caracteres que aparecen desarrollamos un pequeño código en python que nos permita contarlos. Nos descargamos el fichero a local como game-frame.js y ejecutamos este script:

```
characterCounts = {}

line_number = 0

with open('game-frame.js') as infile:

    words = 0

    characters = 0

    for line in infile:

        line_number = line_number +1

        characters = list(line)

        for character in characters:

            if not '' + character in characterCounts:

                characterCounts['' +character] = 0

            characterCounts['' +character] = characterCounts['' +character] + 1

    print(line_number)

    print(characterCounts)
```

En el vemos que la aparición de caracteres en el javascript es la siguiente

```
'[': 70075,
']': 70075,
'(': 13033,
')': 33268,
'+': 60298,
')': 13033,
'\n': 1
```

El código es raro, solamente 6 caracteres distintos,. Tenemos dos posibilidades, un código javascript raro o un texto cifrado, buscamos en google algo que se le parezca para ver qué nos dice:

Google javascript language 6 characters

All Images Videos News Shopping More Settings Tools

About 1,060,000,000 results (0.37 seconds)

JSFuck - Write any JavaScript with 6 Characters: []()!+
www.jsfuck.com/ ▾
JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to execute code.

JSFuck - Wikipedia
https://en.wikipedia.org/wiki/JSFuck ▾
JSFuck is an esoteric programming style of JavaScript, where code is written using only six characters: [,], (,), !, and +. The name is derived from Brainfuck, an esoteric programming language ...
Encoding methods · Numbers · Letters · Character table

JSfuck: write any JavaScript with 6 Characters – So Long, and Thanks ...
https://www.andreafortuna.org/.../jsfuck-write-any-javascript-with-6-characters/ ▾
Oct 19, 2016 - Yes, only []()!+. JSFuck is an esoteric programming language with a very limited set of characters: (,), [], +, !. The name is derived from ...

JSFuck - Esolang
https://esolangs.org/wiki/JSFuck ▾
Aug 23, 2015 ~ JSFuck is an esoteric subset of the JavaScript language that uses only six distinct characters in the source code. The characters are +, !, (,), [...

GitHub - aemkei/jsfuck: Write any JavaScript with 6 Characters: []()!+
https://github.com/aemkei/jsfuck ▾
JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to write and execute code.
Missing: language

A Javascript journey with only six characters | Hacker News
https://news.ycombinator.com/item?id=12666361 ▾
Oct 8, 2016 - I'm a fan of Javascript. It has proper lambdas, true lexical scope, will soon have TCO, and is a really flexible language. But it's not without its ...

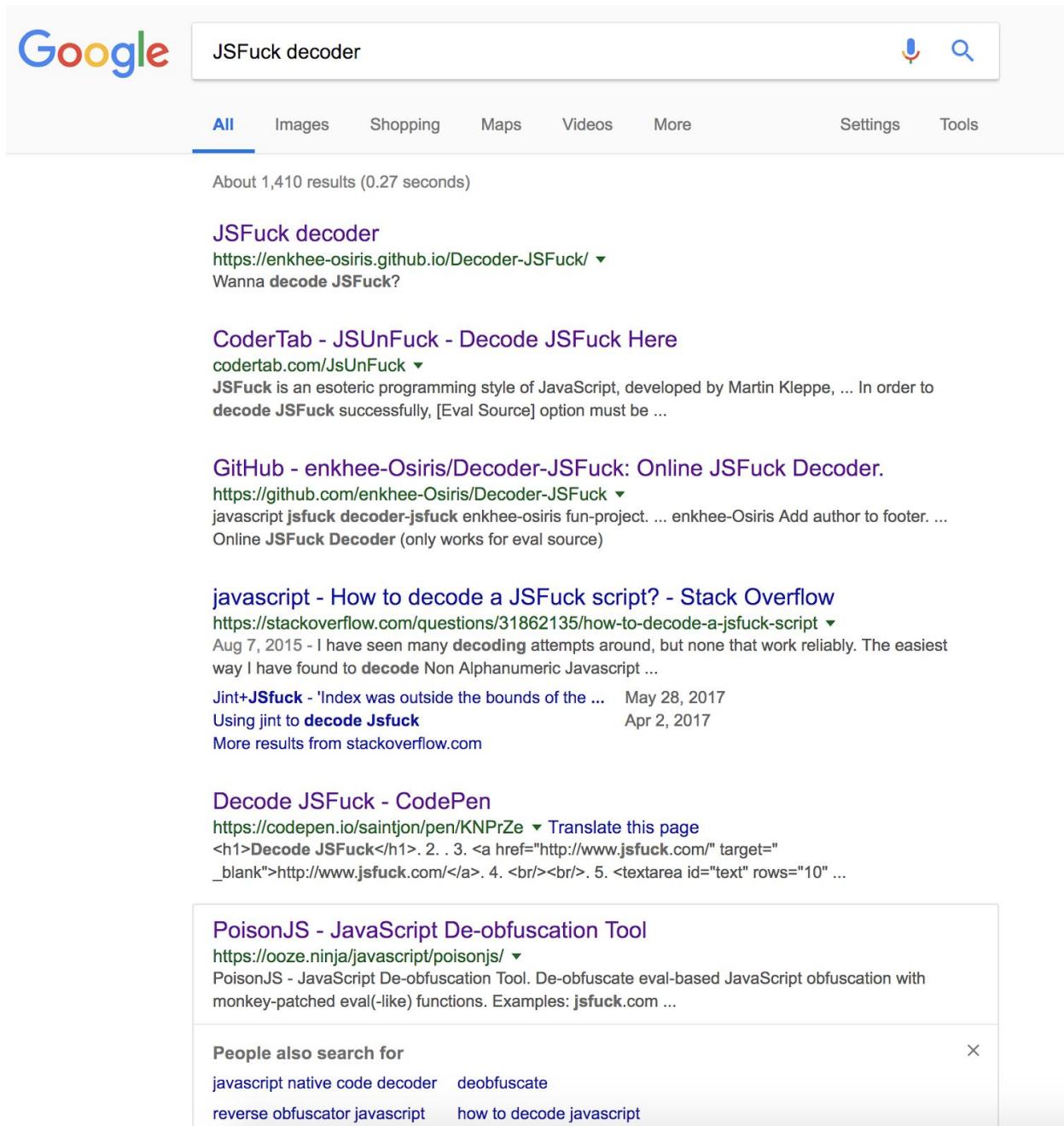
JSFuck - Write any JavaScript with 6 Characters: []()!+ ...
https://www.reddit.com/r/javascript/.../jsfuck_write_any_javascript_with_6_characters/ ▾
Dec 4, 2015 ~ 13 posts · 13 authors

Las primeras búsquedas parecen todas referenciar a una misma idea, JSFuck, veamos que nos dice la wikipedia

"JSFuck is an esoteric programming style of [JavaScript](#), where code is written using only six characters: [,], (,), !, and +. The name is derived from [Brainfuck](#), an [esoteric programming language](#) which also uses a minimalistic [alphabet](#) of only punctuation. Unlike Brainfuck, which requires its own [compiler](#) or [interpreter](#), JSFuck is valid JavaScript code, meaning JSFuck programs can be run in any [web browser](#) or [engine](#) that interprets JavaScript"

Bueno, veamos si hay algo que nos permita obtener el código o parte del mismo en Javascript "normal" para poder leer que es lo que pone en dicho código

Probamos varios de los que nos da la búsqueda de google



The screenshot shows a Google search results page for the query "JSFuck decoder". The results are as follows:

- JSFuck decoder**
<https://enkhee-osiris.github.io/Decoder-JSFuck/> ▾
Wanna decode JSFuck?
- CoderTab - JSUnFuck - Decode JSFuck Here**
codertab.com/JsUnFuck ▾
JSFuck is an esoteric programming style of JavaScript, developed by Martin Kleppe, ... In order to decode JSFuck successfully, [Eval Source] option must be ...
- GitHub - enkhee-Osiris/Decoder-JSFuck: Online JSFuck Decoder.**
<https://github.com/enkhee-Osiris/Decoder-JSFuck> ▾
javascript jsfuck decoder-jsfuck enkhee-osiris fun-project. ... enkhee-Osiris Add author to footer. ...
Online JSFuck Decoder (only works for eval source)
- javascript - How to decode a JSFuck script? - Stack Overflow**
<https://stackoverflow.com/questions/31862135/how-to-decode-a-jsfuck-script> ▾
Aug 7, 2015 - I have seen many decoding attempts around, but none that work reliably. The easiest way I have found to decode Non Alphanumeric Javascript ...
Jint+JSfuck - Index was outside the bounds of the ... May 28, 2017
Using jint to decode Jsfuck Apr 2, 2017
More results from stackoverflow.com
- Decode JSFuck - CodePen**
<https://codepen.io/saintjon/pen/KNPrZe> ▾ Translate this page
<h1>Decode JSFuck</h1>. 2. . 3. http://www.jsfuck.com/. 4.

. 5. <textarea id="text" rows="10" ...
- PoisonJS - JavaScript De-obfuscation Tool**
<https://ooze.ninja/javascript/poisonjs/> ▾
PoisonJS - JavaScript De-obfuscation Tool. De-obfuscate eval-based JavaScript obfuscation with monkey-patched eval(-like) functions. Examples: jsfuck.com ...

Below the search results, there is a "People also search for" section with the following suggestions:

- javascript native code decoder deobfuscate
- reverse obfuscator javascript how to decode javascript

y llegamos al indicado como PoisonJS. Ponemos el código que tenemos, y nos da como resultado:

donde en el último bloque de código podemos leer

PoisonJS - JavaScript De-obfuscation Tool
De-obfuscate eval-based JavaScript obfuscation with monkey-patched eval(-like) functions.
Examples: jsfuck.com javascript2img.com whak.ca

Your example

```
[1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21][22][23][24][25][26][27][28][29][30][31][32][33][34][35][36][37][38][39][40][41][42][43][44][45][46][47][48][49][50][51][52][53][54]
```

De-obfuscate

```
35. return unescape
36. return escape
37. return unescape
38. return escape
39. return unescape
40. return escape
41. return escape
42. return escape
43. return new Date(24000000000)
44. return unescape
45. return escape
46. return unescape
47. return escape
48. return unescape
49. return escape
50. return unescape
51. return escape
52. return unescape
53. return escape
54. parent.postMessage(window.location.toString(), "*"); var originalAlert = window.alert; window.alert = function(s) { parent.postMessage("success", "*"); setTimeout(function() { originalAlert("Enhorabuena, has ejecutado una alerta:\n\n" + s + "\n\nAquí tienes tu password para el zip:\n\nOsLoHaPerDidOaSuPrImo"); }, 50); }
```

mail@ooze.ninja

```
54. parent.postMessage(window.location.toString(), "*"); var originalAlert = window.alert;
window.alert = function(s) { parent.postMessage("success", "*"); setTimeout(function() {
originalAlert("Enhorabuena, has ejecutado una alerta:\n\n" + s + "\n\nAquí tienes tu
password para el zip:\n\nOsLoHaPerDidOaSuPrImo"); }, 50); }
```

Ya tenemos la clave del Zip. "OsLoHaPerDidOaSuPrImo" , lo siento por Oslo, pero, ¡Vamos a por el Zip!

Fichero de audio dentro del Zip

Al descomprimir el zip, obtenemos un fichero de audio , episodio2.wav, vamos a ver que información útil nos dan unas herramientas de información de archivos de audio.

Empezamos con la primera, ffmpeg, que como dice la wikipedia “FFmpeg is a free software project, the product of which is a vast software suite of libraries and programs for handling video, audio, and other multimedia files and streams”

```
$ ffmpeg -v info -i episodio2.wav -f null -  
ffmpeg version 3.4.2 Copyright (c) 2000-2018 the FFmpeg developers  
built with Apple LLVM version 9.0.0 (clang-900.0.39.2)  
configuration: --prefix=/usr/local/Cellar/ffmpeg/3.4.2 --enable-shared  
--enable-pthreads --enable-version3 --enable-hardcoded-tables --enable-avresample  
--cc=clang --host-cflags= --host-ldflags= --disable-jack --enable-gpl  
--enable-libmp3lame --enable-libx264 --enable-libxvid --enable-opencl  
--enable-videotoolbox --disable-lzma  
  
libavutil      55. 78.100 / 55. 78.100  
  
libavcodec     57.107.100 / 57.107.100  
  
libavformat    57. 83.100 / 57. 83.100  
  
libavdevice    57. 10.100 / 57. 10.100  
  
libavfilter     6.107.100 /  6.107.100  
  
libavresample   3.  7.  0 /  3.  7.  0  
  
libswscale       4.  8.100 /  4.  8.100  
  
libswresample   2.  9.100 /  2.  9.100  
  
libpostproc     54.  7.100 / 54.  7.100  
  
Guessed Channel Layout for Input Stream #0.0 : mono  
  
Input #0, wav, from 'episodio2.wav':  
  
  Duration: 00:00:10.18, bitrate: 176 kb/s  
  
    Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 11025 Hz, mono, s16, 176  
    kb/s
```

```
Stream mapping:  
  
Stream #0:0 -> #0:0 (pcm_s16le (native) -> pcm_s16le (native))  
  
Press [q] to stop, [?] for help  
  
Output #0, null, to 'pipe':  
  
Metadata:  
  
encoder : Lavf57.83.100  
  
Stream #0:0: Audio: pcm_s16le, 11025 Hz, mono, s16, 176 kb/s  
  
Metadata:  
  
encoder : Lavc57.107.100 pcm_s16le  
  
size=N/A time=00:00:10.17 bitrate=N/A speed=2.24e+03x  
  
video:0kB audio:219kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
```

No aparece mucha información útil, que nos pueda dar pistas para ver qué contiene. La segunda herramienta que utilizamos es exiftool , “ExifTool is a free and open-source software program for reading, writing, and manipulating image, audio, video, and PDF metadata”. El resultado es el siguiente:

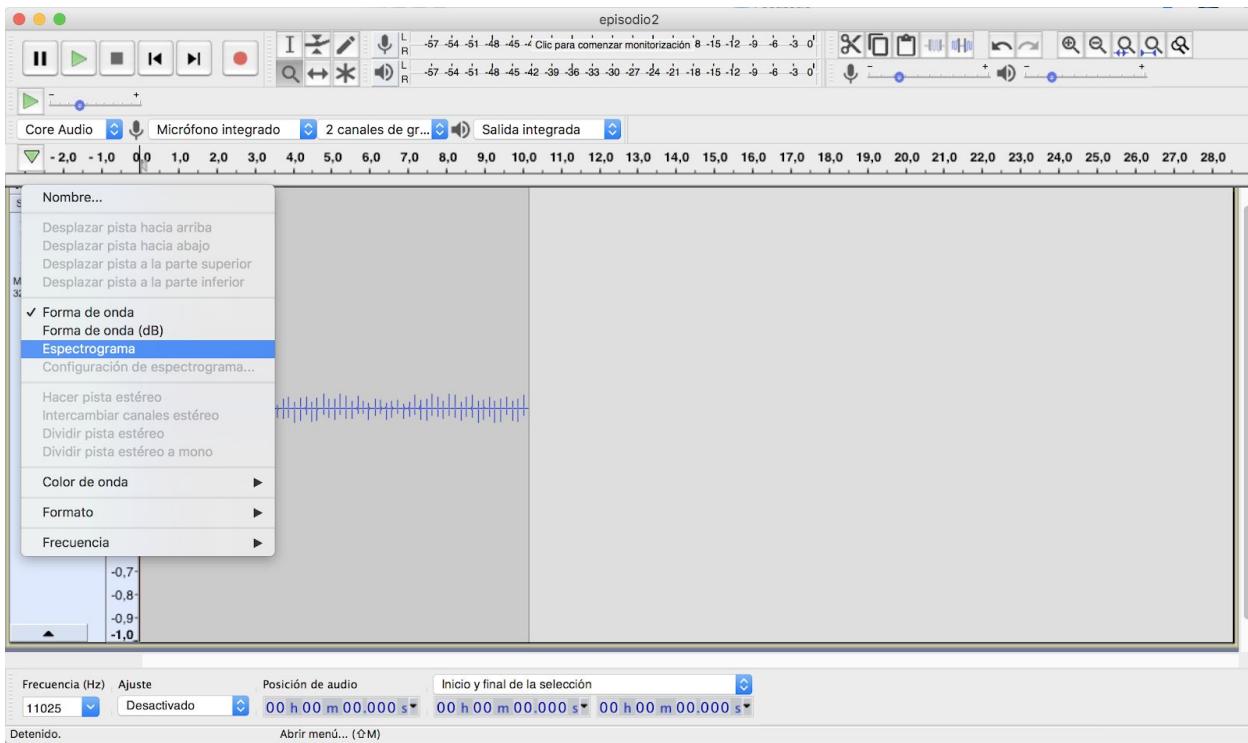
```
$ exiftool episodio2.wav  
  
ExifTool Version Number : 11.01  
  
File Name : episodio2.wav  
  
Directory : Downloads  
  
File Size : 219 kB  
  
File Modification Date/Time : 2018:06:14 12:13:04+02:00  
  
File Access Date/Time : 2018:06:16 14:00:08+02:00  
  
File Inode Change Date/Time : 2018:06:16 13:35:18+02:00
```

```
File Permissions          : rw-r--r--
File Type                : WAV
File Type Extension      : wav
MIME Type                : audio/x-wav
Encoding                 : Microsoft PCM
Num Channels             : 1
Sample Rate              : 11025
Avg Bytes Per Sec       : 22050
Bits Per Sample          : 16
Duration                : 10.18 s
```

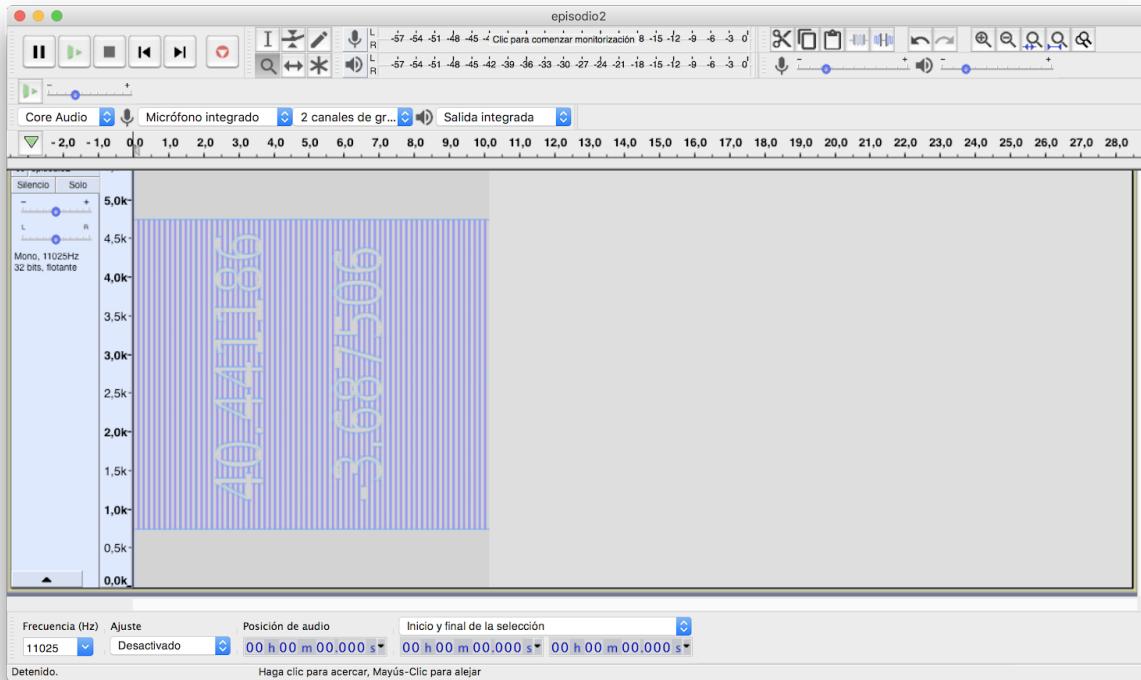
Como vemos, la información que obtenemos tampoco nos ayuda mucho con el fichero. Lo más útil que observamos es que solamente tiene un canal de audio() por lo que podemos descartar determinados métodos de esteganografía que ocultan la información en la diferencia entre dos o más canales en el audio. Vamos a proceder a oír el fichero, lo que escuchamos nos deja oido-platicos ya que no es una música o una voz como suelen tener este tipo de ficheros sino una serie de ruidos que no se identifican con ningún mensaje. La primera idea que nos surge es que se pueda tratar de código morse, muy utilizado en los CTFs pero no parece que exista una distinción clara entre sonidos que reflejan un punto y sonidos que reflejan una raya, así que buscamos en google “steganography audio only noise” y en su tercera entrada encontramos un detalle interesante

The screenshot shows a Google search results page. The search query is "steganography audio only noise". The results are filtered by "All" and show approximately 121,000 results in 0.40 seconds. The first result is a PDF titled "Audio Steganography Using High Frequency Noise Introduction" from Semantic Scholar, with a link to the document and a note that it has been cited by 4 other articles. The second result is a link to "Audio Steganography Research | Scientific method - ResearchGate" with a snippet about hiding data in audio files. The third result is a blog post titled "Basic methods of Audio Steganography (spectrograms) - solusipse" with a snippet about hidden data in audio tracks. A sidebar titled "People also search for" lists related terms: audio steganography decoder, spectrogram steganography, mp3 steganography linux, audio steganography python, wav steganography ctf, and mp3 steganography ctf.

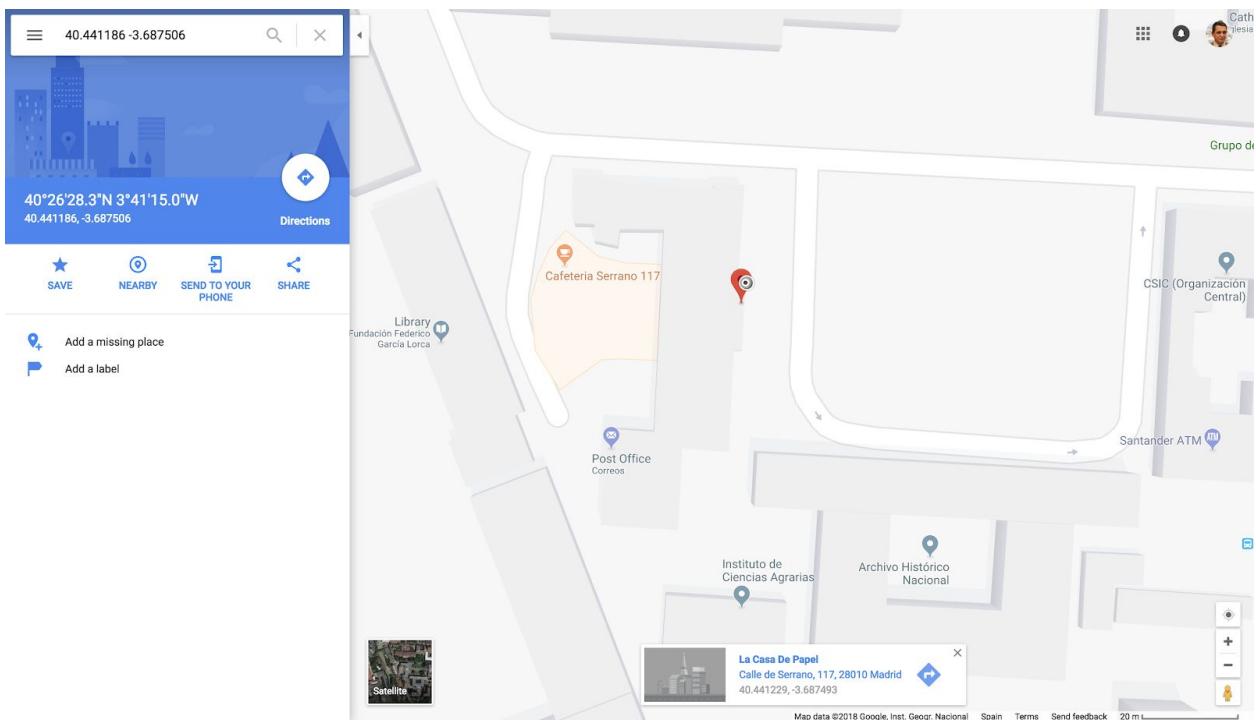
Buscamos un programa que nos permita mostrar estos espectrogramas , ya que no encontramos el programa al que hacen referencia imageEncode para mac desde una fuente fiable. Con otra búsqueda en Google vemos que Audacity nos permite mostrar dichos espectrogramas. Probamos suerte pulsando en el menú de la flecha desplegable de la columna izquierda donde aparece el nombre del archivo y seleccionamos espectrograma



y encontramos unos valores numéricos que parecen coordenadas geográficas



¿Que habrá en el punto 40.441186 -3.687506? Miramos en Google Maps



¡Vaya! Son las coordenadas de la casa de papel.

A partir de aquí miramos los comentarios que aparecen en google sobre ese lugar, en total son 17

[← All reviews](#) [X](#)

WRITE A REVIEW



Sort by: Newest ▾



angel campos garcia



★★★★★ 3 days ago



Antonio Becker

172 reviews · 25 photos



★★★★★ 5 days ago
(Translated by Google) Horrible place very hermetic

to the North Korean

(Original)

Horrible lugar muy herméticos a lo Norcoreano

Like Share



Cyril M

Local Guide · 4 reviews · 111 photos



★★★★★ a week ago
(Translated by Google) We can not enter. Just make

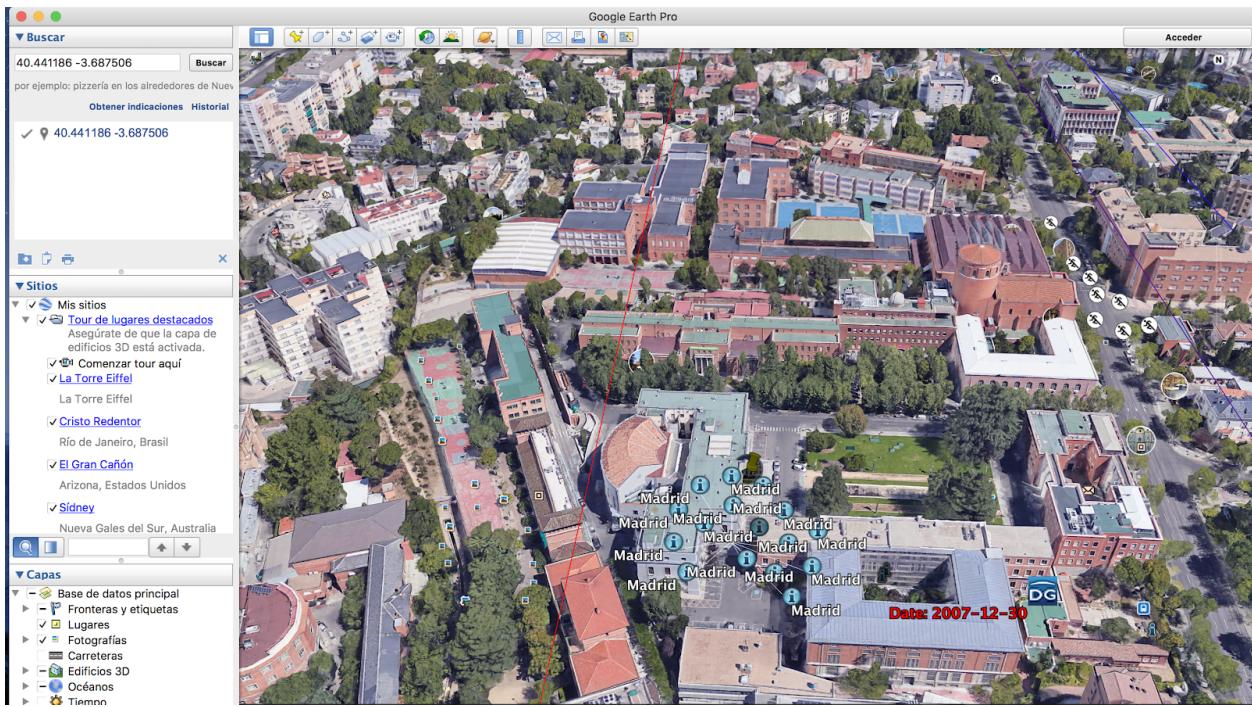
a quick photo through the grids, if you are lucky

enough not to have a truck in front.

(Original)

On ne peut pas entrer. Juste faire une photo rapide à

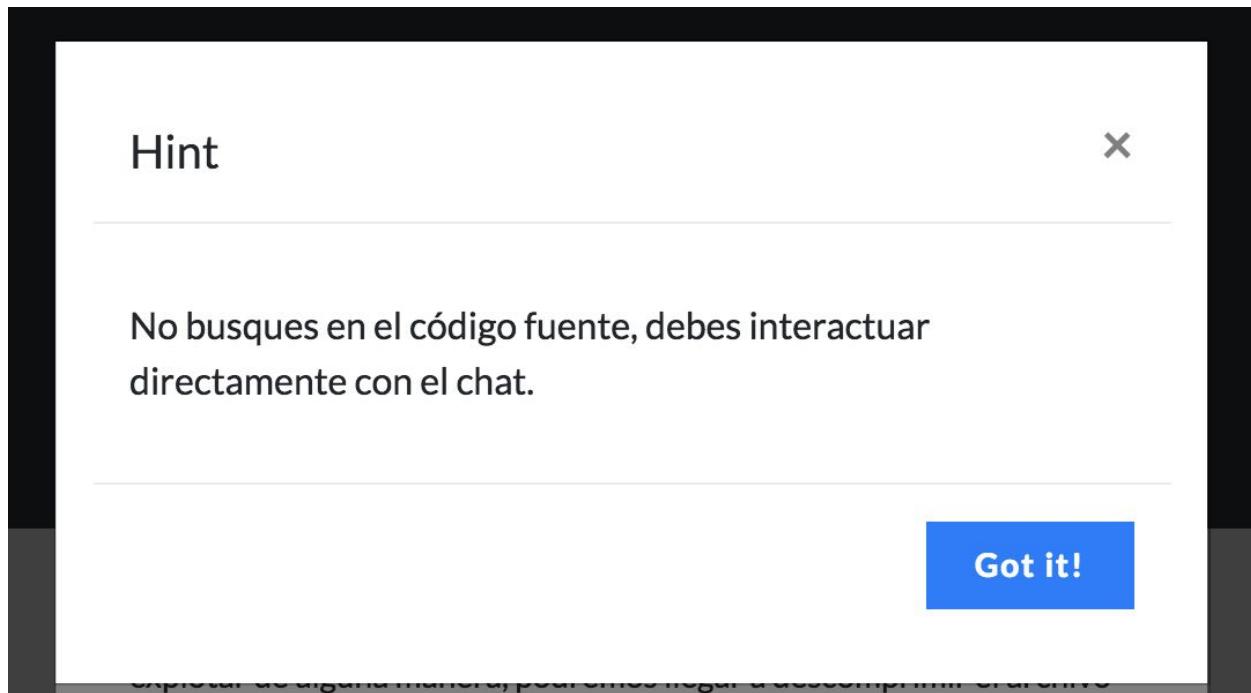
Los leemos todos, el de corea del norte parece que pueda ser una pista, miramos a todos los usuarios y sus comentarios en otras localizaciones y nada, no parece haber ninguna pista que nos ayude a obtener el MD5 de la flag. Como las coordenadas de la imagen aparecen rotadas 90 grados a la izquierda, pensamos que esto también pueda ser una pista, pero si giramos las coordenadas que tenemos 90 grados en el mapa tampoco parece que haya nada significativo. Miramos en Google Earth para ver si hay algo por allí pero tampoco encontramos nada.



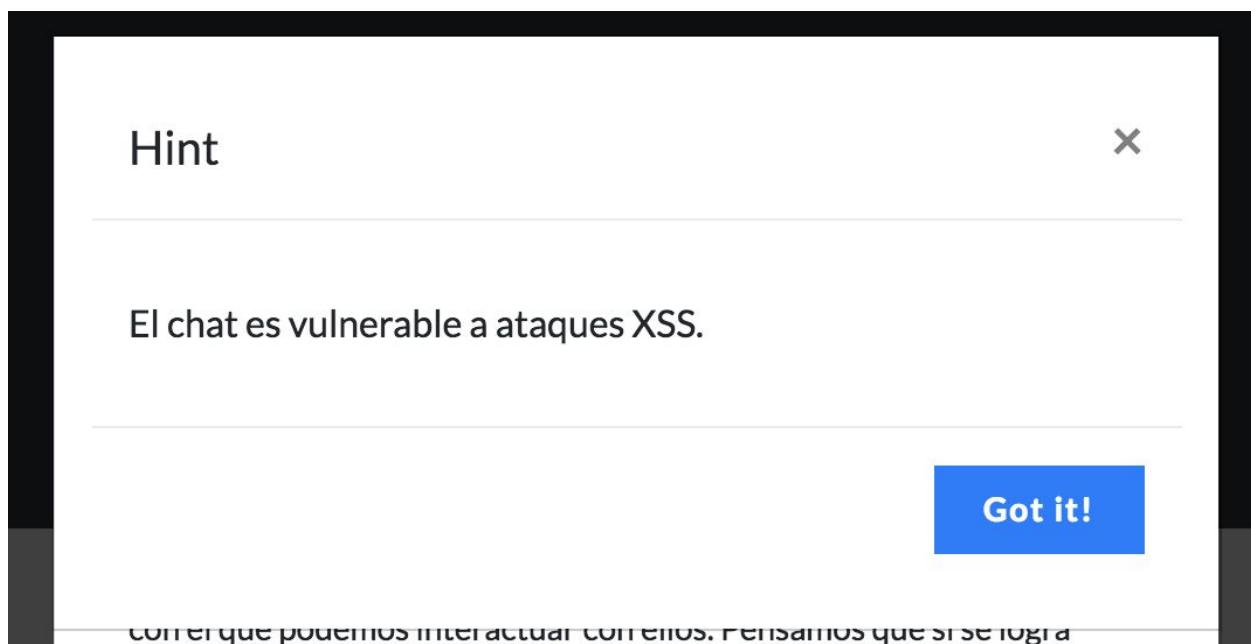
Vuelta al inicio. Segunda manera de obtener la contraseña del zip

Tras mucho darle vueltas a las coordenadas geográficas que hemos encontrado, volvemos al principio, aún tenemos dos pistas que no hemos utilizado, gastaremos unos cuantos puntos pero a lo mejor merecen la pena. Probemos con ello.

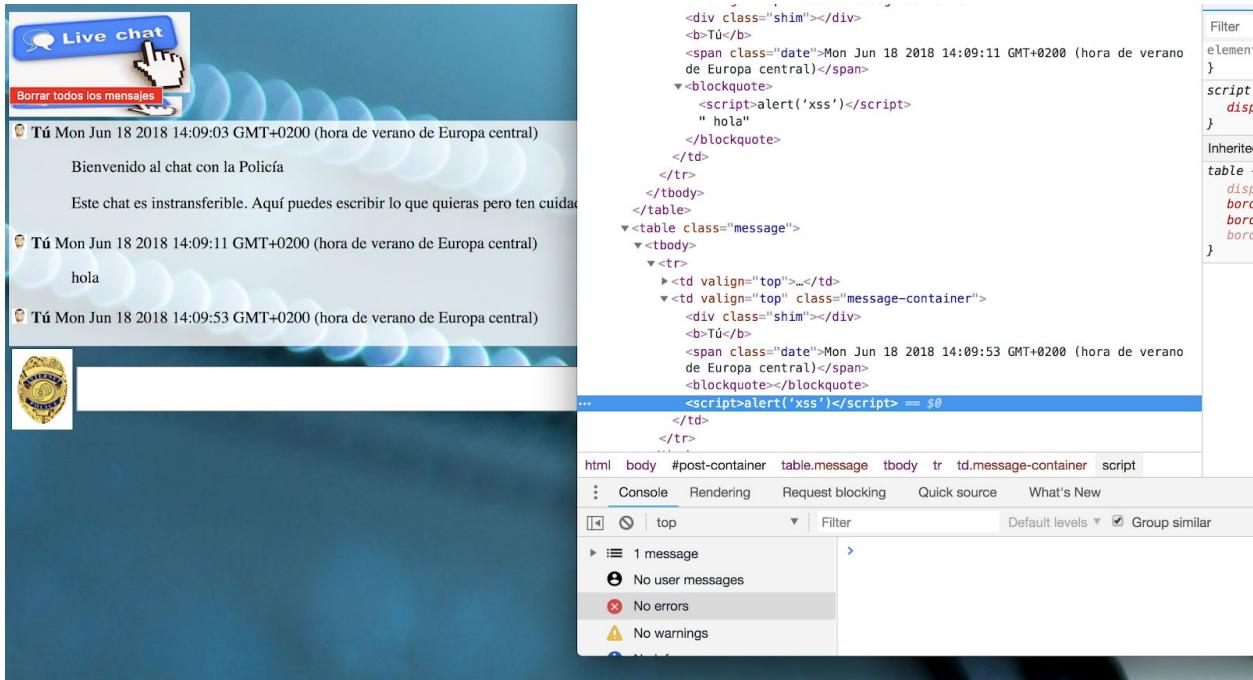
La primera pista nos dice



Justo hemos hecho eso, a lo mejor todos los pasos anteriores nos han llevado a un callejón sin salida que nos han preparado los organizadores, veamos que dice la segunda pista



Bueno pues probaremos algunas de las técnicas de XSS sobre el chat. Lo primero intentar un tag html <script> con el clásico <script>alert('xss')</script> pero parece que algo impide su ejecución



The screenshot shows a live chat interface with a blue header bar. A cursor is hovering over a red button labeled "Borrar todos los mensajes". The main area displays a conversation:

Tú Mon Jun 18 2018 14:09:03 GMT+0200 (hora de verano de Europa central)
Bienvenido al chat con la Policía
Este chat es intransferible. Aquí puedes escribir lo que quieras pero ten cuidado.

Tú Mon Jun 18 2018 14:09:11 GMT+0200 (hora de verano de Europa central)
hola

Tú Mon Jun 18 2018 14:09:53 GMT+0200 (hora de verano de Europa central)

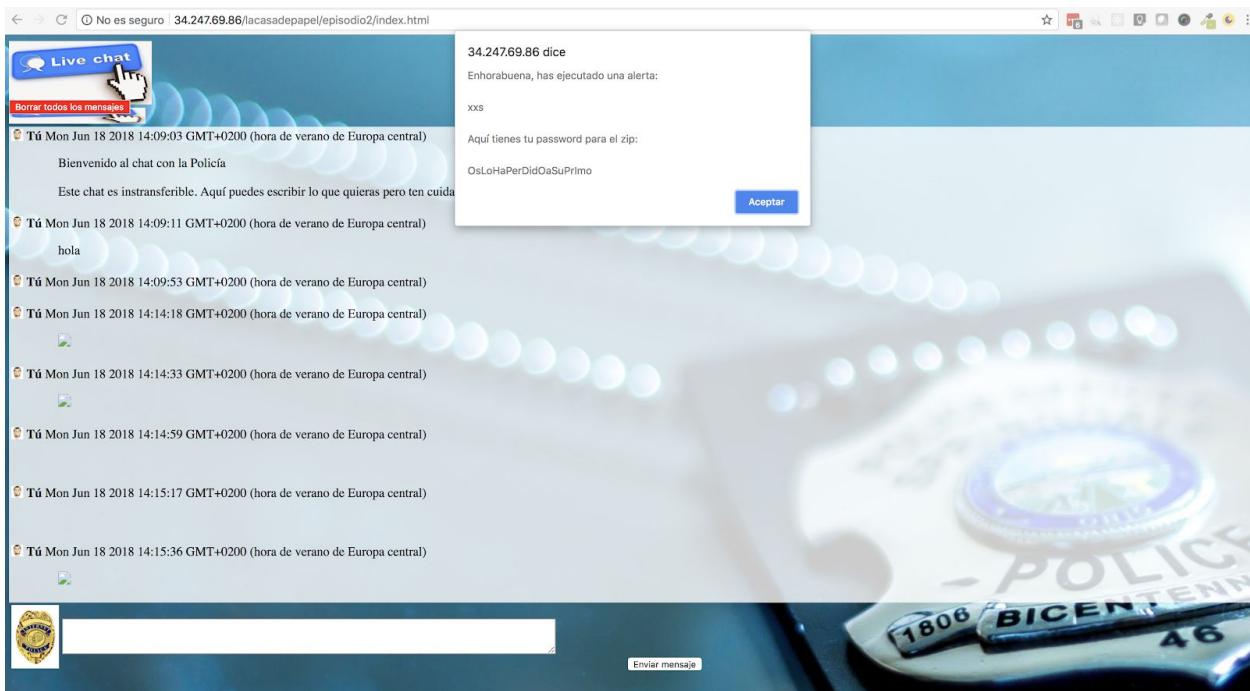
The message "hola" contains a script tag: <script>alert('xss')</script>. This tag is highlighted in the browser's developer tools, specifically in the "Elements" tab under the "script" section. The developer tools also show the full HTML structure of the message, including the <td> and <td.message-container> elements.

Probamos otra serie de técnicas , como las descritas en

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Al final con el mensaje

y pasando el ratón por encima de la imagen rota obtenemos el mensaje.



Desgraciadamente no hay información adicional de la que ya habíamos obtenido.

Con las coordenadas, hasta la resolución.

Después de unas cuantas horas intentando obtener algo más de las dichosas coordenadas al final la solución era más sencilla de lo que parecía. El MD5 que buscaba era el MD5 de estas coordenadas. Después de probar con un retorno de carro entre ellas (ya que en la imagen del wav aparecían en dos líneas) en formato windows , linux, con comas, con puntos, la solución era el MD5 de las mismas separadas por un espacio y la flag por lo tanto era

UAM{9bbf31b30acd21df0d35a4d8333b235e}

P.D https://es.wikipedia.org/wiki/Navaja_de_Ockham

La navaja de Ockham (a veces escrito Occam u Ockam), principio de economía o principio de parsimonia (lex parsimoniae), es un principio metodológico y filosófico atribuido al fraile

franciscano, filósofo y lógico escolástico Guillermo de Ockham (1280-1349), según el cual:

En igualdad de condiciones, la explicación más sencilla suele ser la más probable.