Una-al-mes: Episodio#2 - Hispasec

Enunciado CTF:

Mientras estábamos dentro de la caja fuerte, la policía ha podido entrar en el sistema informático de la fábrica. Nos ha abierto un chat "seguro" con el que podemos interactuar con ellos. Pensamos que si se logra explotar de alguna manera, podremos llegar a descomprimir el archivo que tiene la flag.

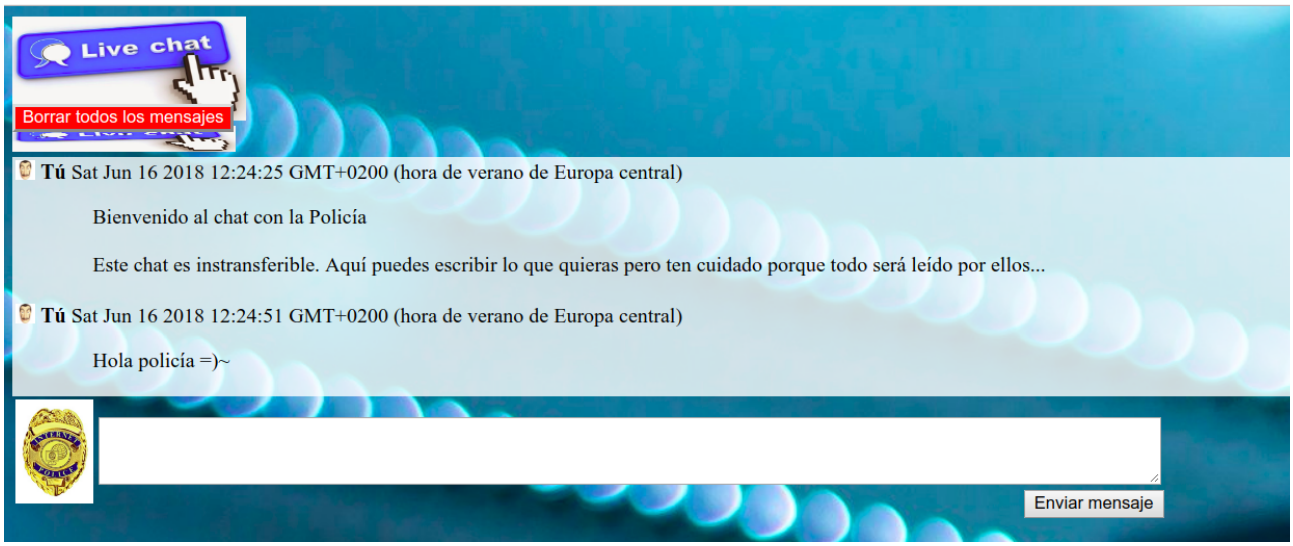
Chat con la Policía: http://34.247.69.86/lacasadepapel/episodio2/index.html

También hay un archivo episodio2.zip que podemos descargar.

Resolución:

Accedemos a la URL que nos facilita el reto:
http://34.247.69.86/lacasadepapel/episodio2/index.html

Vemos que se trata de un chat para comunicarnos con la policía:



Analizamos el código fuente de la página “index.html” y dos archivos “post-store.js” y “game-frame.js”. Si interactuamos con el chat y analizamos el tráfico, vemos que no se manda nada al servidor, es decir, que los mensajes que escribimos en el chat no se envían. Todos los mensajes se gestionan con el javascript de la web en nuestro propio navegador.

Por lo tanto nos centramos en buscar el texto o el evento que haga que el javascript de la web nos devuelva alguna pista.

Al tratarse de un cuadro de texto en el que el usuario puede introducir un payload que luego será ejecutado y renderizado por nuestro navegador, podemos intentar algún tipo de XSS (Cross-Site Scripting).

Como no hay interacción con el servidor, descartamos el tipo reflejado y persistente y nos centramos en el DOM Based, que consiste en atacar al código que se está ejecutando en nuestro navegador.

Podemos ver cómo describe este tipo de vulnerabilidad OWASP:

DOM Based XSS (or as it is called in some texts, “type-0 XSS”) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client side script, so that the client side code runs in an “unexpected” manner. That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

Probamos escribiendo en el chat este payload y le damos al botón “enviar mensaje” sin resultado positivo:
`<script>alert(1);</script>`

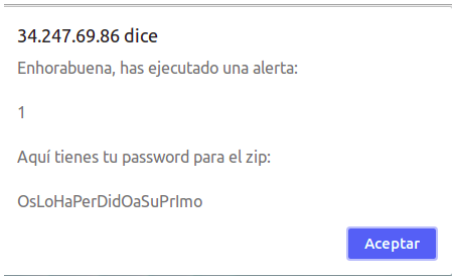
Pero con cualquiera de estos dos payloads sí conseguimos que se produzcan eventos en nuestro código con resultados positivos:

```
<img src="" onerror="alert(1)">
```

o también:

```
<input type=”text” onmouseover="alert(1)">
```

De tal forma que conseguimos ejecutar un modal con una alerta que nos muestra la contraseña del archivo ZIP:



El password es `OsLoHaPerDidOaSuPrimo`

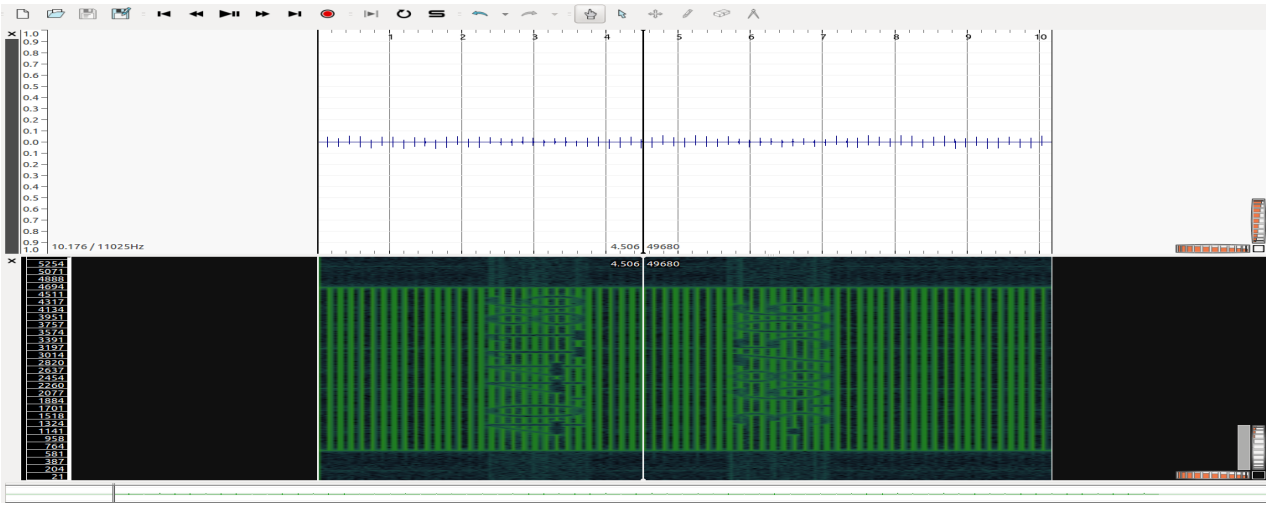
Descomprimimos el ZIP y obtenemos el archivo “episodio2.wav”.

Al escucharlo reproduce una serie de tonos que no nos dan ninguna pista.

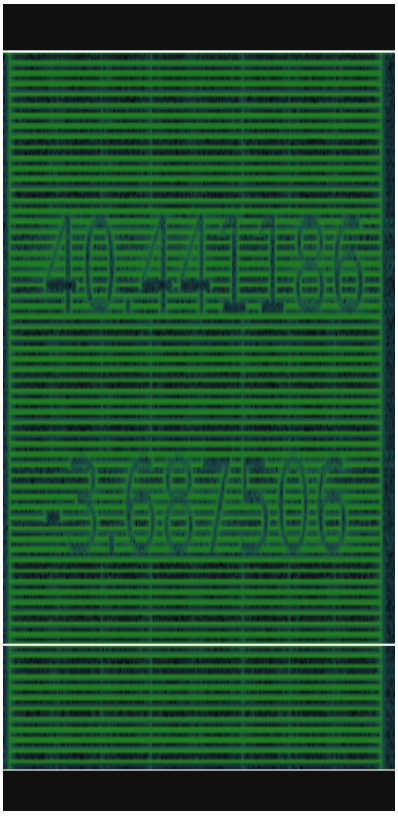
Lo atacamos con algunas herramientas de estaganografía para archivos de sonido por si oculta alguna información, pero no obtenemos nada:

- AudioStego: `hideme episodio2.wav -f && cat output.txt`
- Steghide: `steghide extract -sf episodio2.wav`
- Con un editor hexadecimal como Bless.

Otra de las técnicas de estaganografía en archivos de audio es ocultar información en el espectro de sonido. Para poder analizar el espectro del archivo, podemos usar una herramienta llamada Sonic Visualizer. Pulsando en el menú “Pane > Add Spectogram” vemos el espectro del archivo:

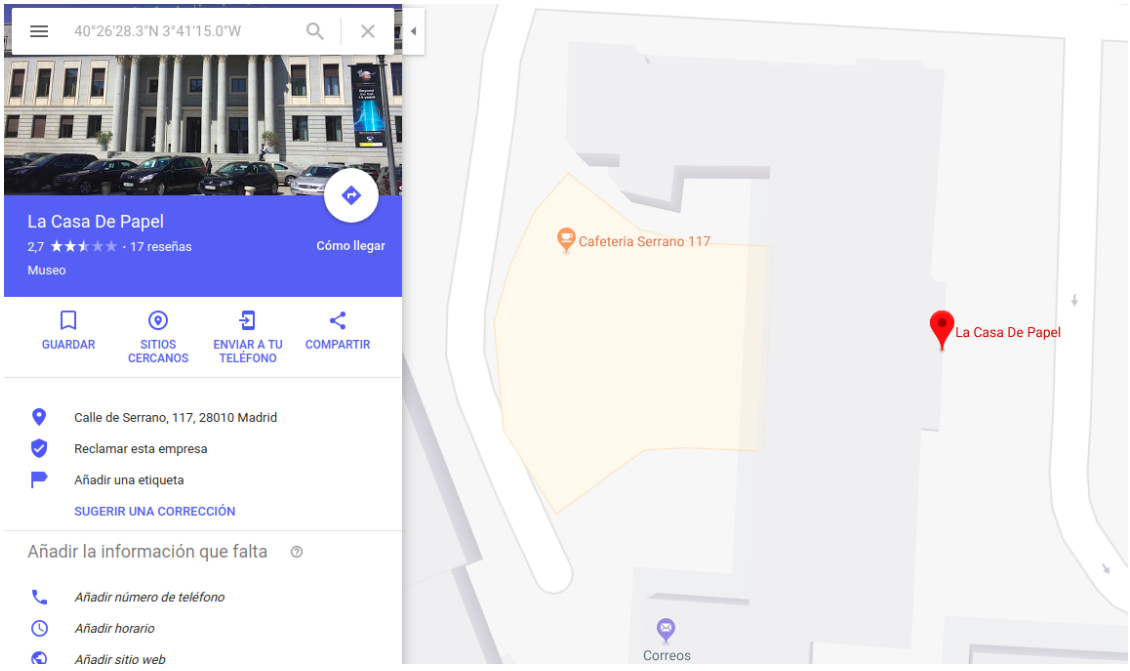


Parece que hay un mensaje oculto. Si giramos la imagen del espectro vemos una coordenadas:



Las coordenadas son: `40.441186  -3.687506`

Si las buscamos en Google Maps llegamos a “La Casa de Papel”.



Como el flag es un md5, probamos a obtener el hash del texto “La Casa De Papel”, pero no es el hash correcto.

Si probamos a realizar el hash de las coordenadas con alguna herramienta online obtenemos:

```
40.441186 -3.687506 → 9bbf31b30acd21df0d35a4d8333b235e
```

Siendo el flag: `UAM{9bbf31b30acd21df0d35a4d8333b235e}`

Rafa Martos
@elbuenodefali