Informe Laboratorio 4

Sección 02

Jorge Toro Macías
e-mail: jorge.toro1@mail.udp.cl

Junio de 2024

Índice

1. Descripción de actividades

Para este laboratorio, deberá utilizar Tampermonkey y la librería CryptoJS (con SRI) para lograr obtener los mensajes que le está comunicando su informante. En esta ocasión, su informante fue más osado y se comunicó con usted a través de un sitio web abierto a todo el público https://cripto.tiiny.site/.

Sólo un ojo entrenado como el suyo logrará descifrar cuál es el algoritmo de cifrado utilizado y cuál es la contraseña utilizada para lograr obtener la información que está oculta.

1. Desarrolle un plugin para tampermonkey que permita obtener la llave para el descifrado de los mensajes ocultos en la página web. La llave debe ser impresa por la consola de su navegador al momento de cargar el sitio web. Utilizar la siguiente estructura:

   - La llave es: KEY

2. En el mismo plugin, se debe detectar el patrón que permite identificar la cantidad de mensajes cifrados. Debe imprimir por la consola la cantidad de mensajes cifrados. Utilizar la siguiente estructura: Los mensajes cifrados son: NUMBER

3. En el mismo plugin debe obtener cada mensaje cifrado y descifrarlo. Ambos mensajes deben ser informados por la consola (cifrado espacio descifrado) y además cada mensaje en texto plano debe ser impreso en la página web.

   El script desarrollado debe ser capaz de obtener toda la información del sitio web (llave, cantidad de mensajes, mensajes cifrados) sin ningún valor forzado. Para verificar el correcto funcionamiento de su script se utilizará un sitio web con otro texto y una cantidad distinta de mensajes cifrados. Deberá indicar la url donde se podrá descargar su script.

   Un ejemplo de lo que se debe visualizar en la consola, al ejecutar automáticamente el script, es lo siguiente:

2. Desarrollo (Parte 1)

2.1. Detecta el cifrado utilizado por el informante

Después de probar con varios sitios de descifrado se logró detectar que el cifrado utilizado por el informante es el de 3DES con input de texto en formato base64. Esto se aprecia claramente a continuación en la Figura 1.

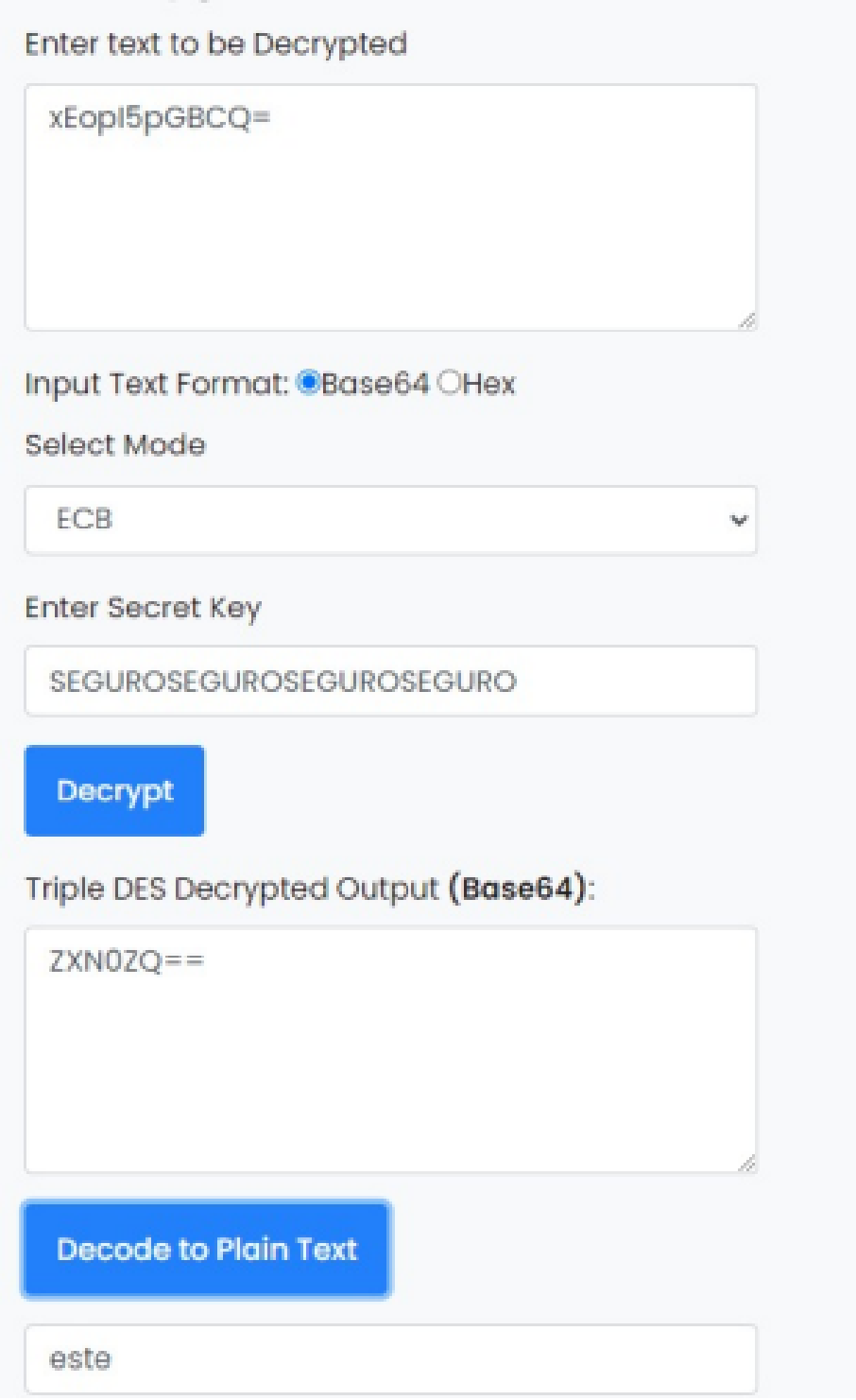

Figura 1: Cifrado utilizado por el informante.

2.2. Logra que el script solo se gatille en el sitio usado por el informante

En los parámetros del script desarrollado en TamperMonkey se indica el sitio sobre el cual actuará el script. Obsérvese en la Figura 2.

Figura 2: Parámetros script de TamperMonkey.

Figura 3: Identificación de sitio sobre el cual actuará el script.

2.3. Define función que obtiene automáticamente el password del documento

Al analizar el sitio web se logró identificar la repetición secuencial del texto, donde la única letra mayúscula de cada parráfo se encontraba al inicio de éste. Entonces, se hizo la prueba separando todas las mayúsculas del resto del texto y juntándolas, dando así resultado el string "SEGUROSEGUROSEGUROSEGURO", el cual tiene sentido y fue considerado como la llave.

La función que obtiene automáticamente el password del documento es la función utilizada para obtener las mayúsculas del texto de la página concatenadas en un string. Este fragmento de código se puede observar en la Figura 4.

```
function imprimirMayusculas() {
    const textoEnPagina = document.body.innerText;
    const letrasMayusculas = textoEnPagina.match(/[A-Z]/g);
    if (letrasMayusculas) {
        const key = letrasMayusculas.join('');
        console.log("La llave es:", key);
    }
}
```

Figura 4: Función implementada para identificar la password.

2.4. Muestra la llave por consola

Como se indicó anteriormente, la llave obtenida es "SEGUROSEGUROSEGUROSEGURO", y es impresa por consola tal como se observa en la Figura 5.

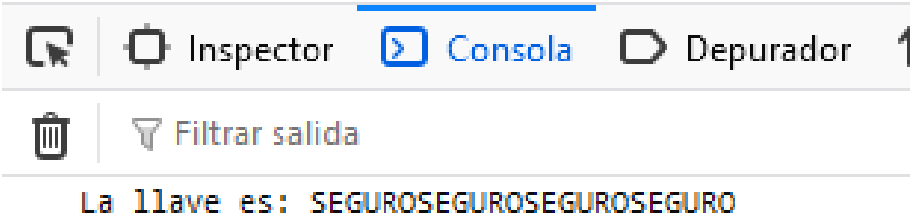

Figura 5: Llave impresa por consola.

3. Desarrollo (Parte 2)

3.1. Reconoce automáticamente la cantidad de mensajes cifrados

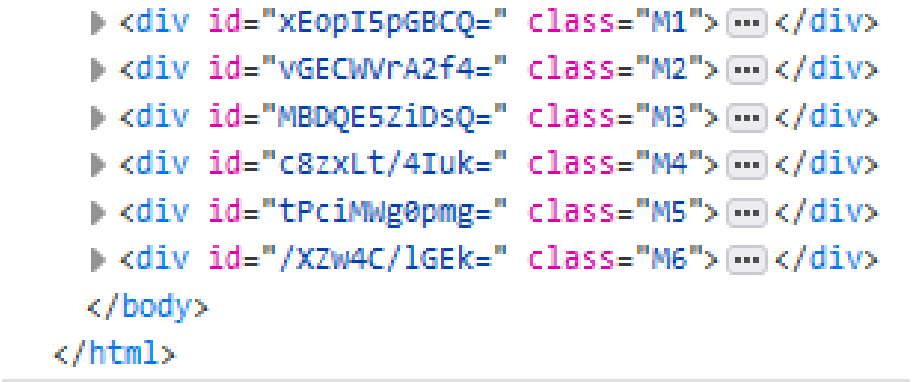

Figura 6: Identificación de los mensajes cifrados.

Al analizar el código fuente de la página web con la función de Inspeccionar elemento se puede observar que existen seis etiquetas de tipo div, con un id en forma de código (Figura 6), lo que los hacía principales candidatos a mensaje cifrado. Los valores de los id fueron puestos a prueba en el descifrador mostrado en la Figura 1 y se logró concluir que efectivamente correspondían a los mensajes cifrados.

Luego, se hizo la automatización del recuento de mensajes cifrados, implementado una función que contara cuántas etiquetas div existían en el código HTML de la página e imprimiera la cantidad, acompañado de los respectivos mensajes cifrados que ocultaban las etiquetas. La implementación de esta función se puede observar en la Figura 7.

```
function contarDivs() {
    const divElements = document.querySelectorAll('div');
    const divCount = divElements.length;
    console.log("Los mensajes cifrados son:", divCount);
}
```

Figura 7: Función implementada para reconocer la cantidad de mensajes cifrados.

3.2. Muestra la cantidad de mensajes por consola

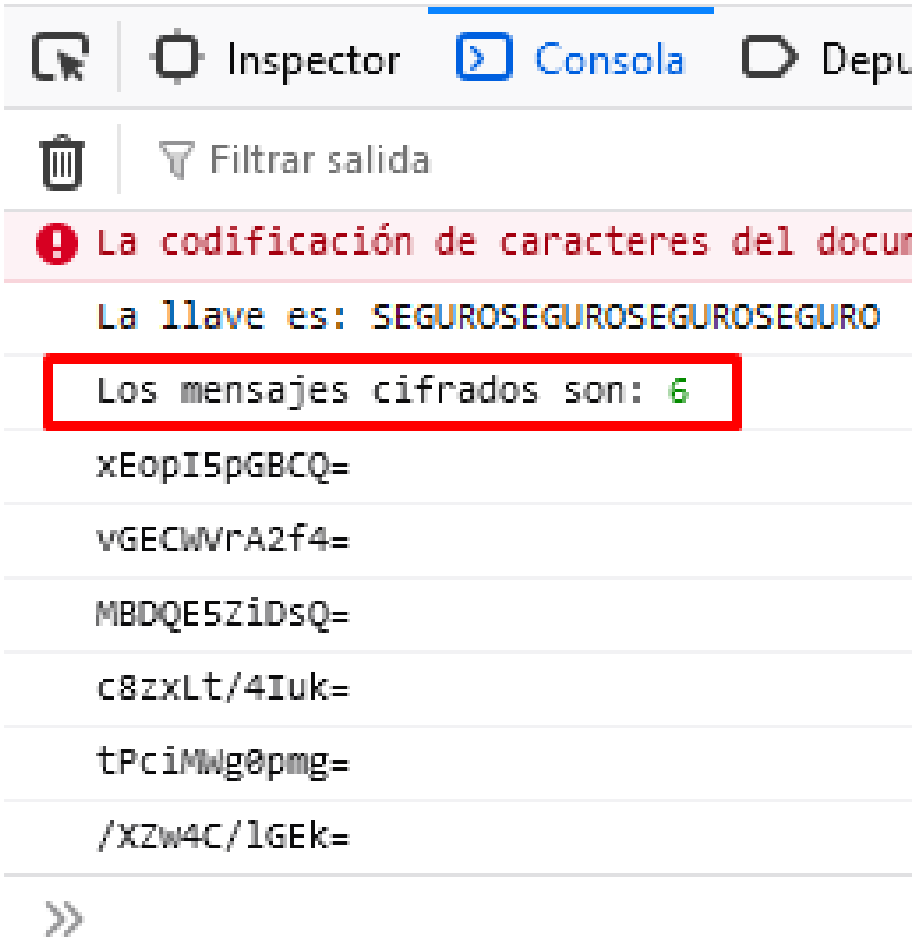

Figura 8: Cantidad de mensajes impresa en consola.

Tal como se observa en la Figura 8, la cantidad de mensajes cifrados es impresa en consola, y corresponde a 6 elementos. Debajo se pueden observar aquellos mensajes.

4. Desarrollo (Parte 3)

4.1. Importa la librería cryptoJS

Figura 9: Web de cdnjs de donde se importa CryptoJS.

```
// @require      https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/crypto-js.min.js#sha512-a+SUDuwNzXDvz4XrIcXHuCf089/iJAoN4lmrXJg18XnduKK6YlDHNRalv4yd1N40OKI80tFidF+rqTFKGPoWFQ==
```

Figura 10: Importación de la librería CryptoJS al script de TamperMonkey.

En la Figura 9 se muestra la web de cdnjs, de donde se importa la librería CryptoJS para su uso en el script de TamperMonkey. La versión seleccionada es la que se puede observar marcada por una casilla en rojo. Luego, en la Figura 10, se muestra cómo es importada en el script, indicado mediante un parámetro '@require'. En la Figura 10 se puede observar

que la url para la importación es más larga que la indicada en la Figura 9. Esto se debe a la adición del SRI Hash, lo que nos lleva al siguiente inciso.

4.2. Utiliza SRI en la librería CryptoJS

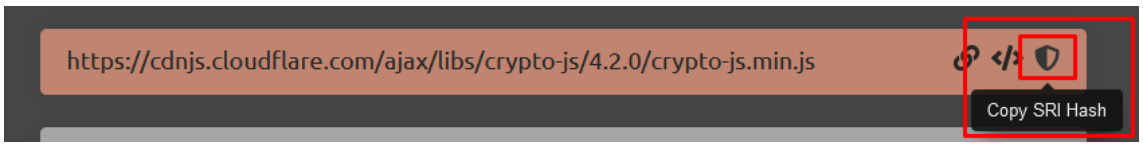

Figura 11: SRI Hash para librería CryptoJS

En la figura 11 se observa cómo es obtenido el SRI Hash para CryptoJS. Este se obtiene de la misma web de cdnjs, en la misma casilla donde se obtiene la url para la importación de la librería. El botón para copiar el SRI Hash se puede observar encasillado en rojo.

4.3. Repercusiones de SRI inválido

Las repercusiones al indicar un SRI inválido, en el contexto de desarrollo de un script para el descifrado, radican principalmente en problemas de funcionamiento del algoritmo. Puede ocurrir un error de carga del recurso o una errónea funcionalidad de este, concluyendo así en la no realización del descifrado, tal como se puede observar en la Figura 12. Para ocasionar el error se reemplazó un solo caracter por otro, dentro de la url de importación de CryptoJS con SRI.

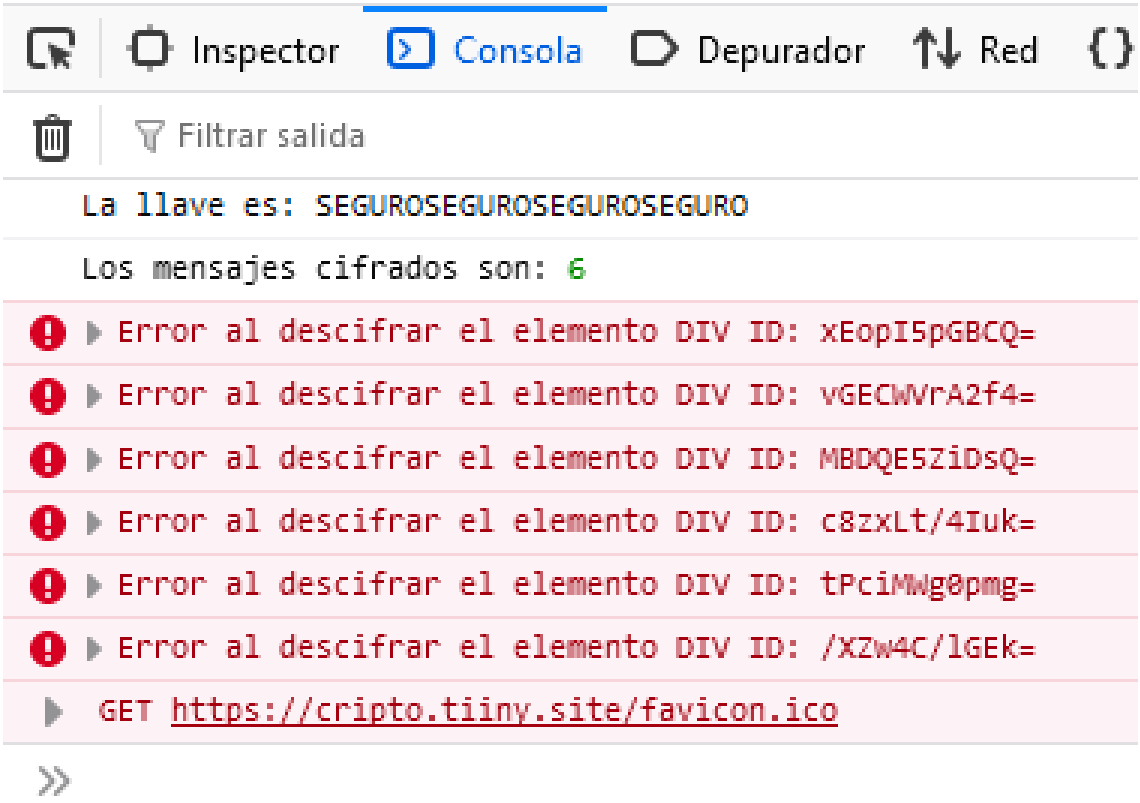

Figura 12: Error en el descifrado.

Además, la carga errónea de recursos puede derivar a una mala experiencia del usuario.

Por otro lado, un SRI inválido en el contexto de desarrollo y aplicación de la seguridad en un sitio web puede ser interpretado como un descuido o incluso como una mala práctica de seguridad por alguien encargado de la revisión de errores.

4.4. Logra decifrar uno de los mensajes

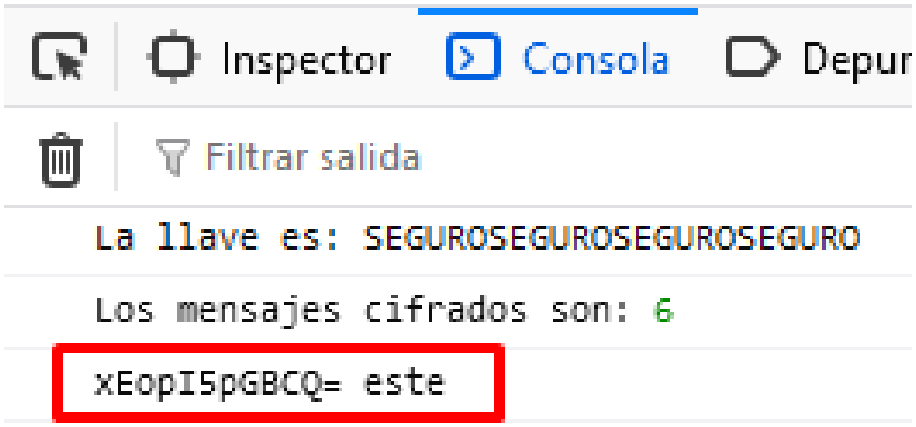

Figura 13: Descifrado de un mensaje.

En la Figura 13 se evidencia la impresión exitosa del mensaje cifrado acompañado de su valor descifrado. En este caso el mensaje cifrado 'xEopI5pGBCQ=' corresponde al string 'este'.

4.5. Imprime todos los mensajes por consola

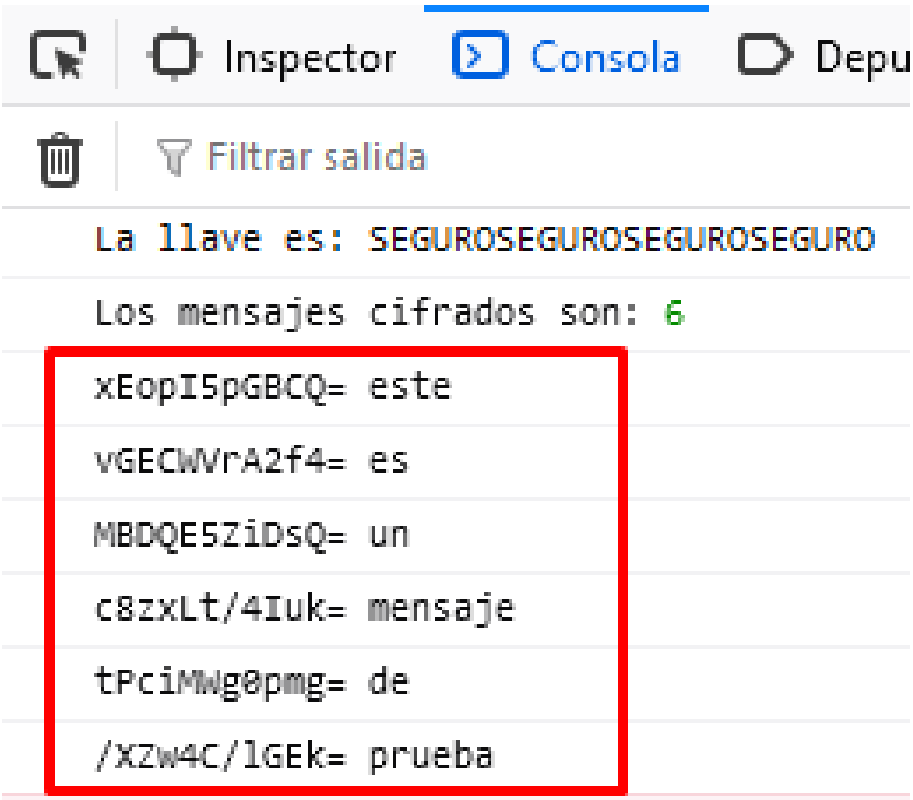

Figura 14: Mensajes descifrados.

En la Figura 14 se pueden observar todos los mensajes descifrados. Juntos, forman la cadena 'este es un mensaje de prueba'.

4.6. Muestra los mensajes en texto plano en el sitio web

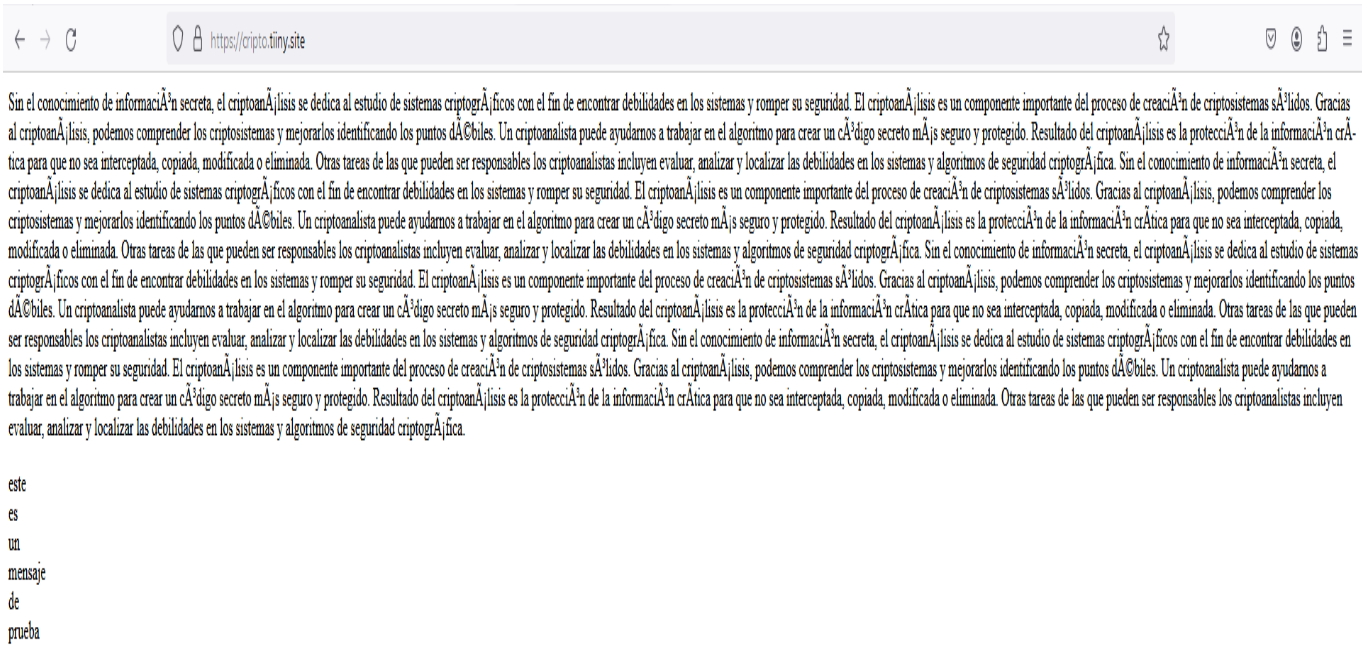

Figura 15: Mensajes en texto plano desplegados en el sitio web.

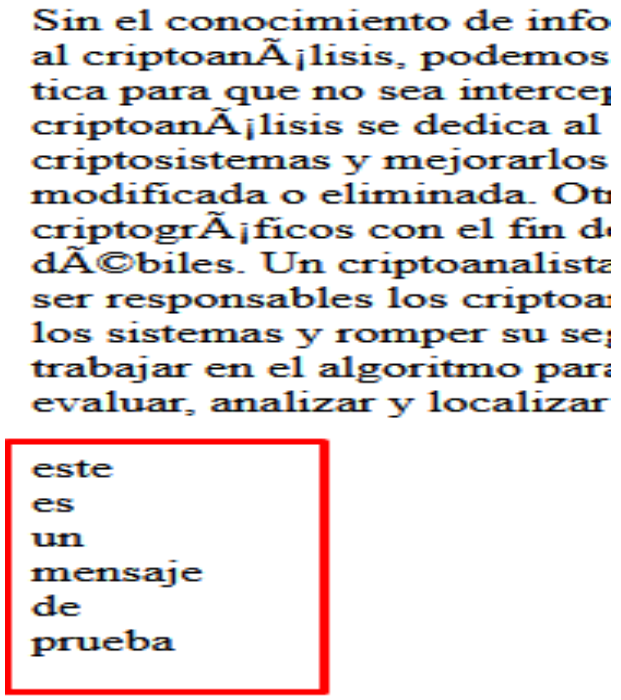

Figura 16: Mensajes en texto plano desplegados en el sitio web. Vista Zoom.

En las figuras 15 y 16 se puede observar cómo han sido agregados los mensajes descifrados en el texto original del sitio web.

4.7. El script logra funcionar con otro texto y otra cantidad de mensajes

4.8. Indica url al código .js implementado para su validación

Script

Conclusiones y comentarios

Issues

La mayoría de los problemas encontrados fueron en la implementación del descifrado utilizando la librería CryptoJS. El primer problema fue encontrar una plataforma de dónde importarla junto al SRI Hash. En primera instancia estaba importándolo desde plataformas como GreasyFork, que no incluían el SRI. Además, por alguna razón, no importaba correctamente esa versión de CryptoJS y por lo tanto no cargaban los recursos, lo que corresponde al segundo problema. En tercer lugar se presentaron problemas al implementar

las funciones de descifrado, ya que en los primeros intentos no se tenía claro el tipo de cifrado, por lo que acababan fallando al desarrollar y probar funciones no correspondientes.

Todos los problemas mencionados fueron resueltos mediante una investigación más a fondo de plataformas más completas para la importación de librerías y el desarrollo de funciones para el descifrado. Fue así como se encontró cdnjs, la cual resolvió el problema del SRI principalmente y el de la mala carga de recursos. Luego, para la implementación de la función de descifrado bastó con identificar el cifrado que estaba siendo utilizado por el informante, y así investigar cómo implementar las funciones correspondientes a ese tipo de cifrado, en este caso 3DES.

El último problema fue al modificar el código para que no estuviese hardcodeado, pues en la primera versión de este se declaraba y se definía la llave unicamente como "SEGUROSEGUROSEGUROSEGURO", por lo que claramente no serviría en otros sitios con el mismo índole de cifrado. Por lo tanto se volvió a editar el código de manera que antes de descifrar volviera a tomar todas las mayúsculas y las concatenara para obtener la llave y trabajar con eso el descifrado, todo dentro de la función destinada a ello.