Informe Laboratorio 4

Sección 2

Cristóbal Romero
e-mail: cristobal.romero_f@mail.udp.cl

Junio 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

El cifrado usado por el informante se trata de esconder sus mensajes a traves de los id de containers escondidos a simple vista en la página, además la llave para poder avanzar con el descifrado está escondido a simple vista en el texto.

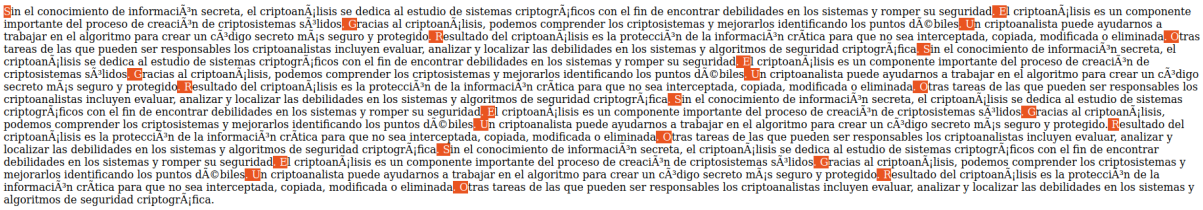

Figura 1: Llave escondida en párrafo de la página.

La llave se esconde mediante cada primera letra de las oraciones presentes en el párrafo, estas se repiten tomando siempres las misma letras un total de 4 veces.

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

Para que el script TamperMonkey solo se use en la página de `https://cripto.tiiny.site` es necesario usar la función match de TamperMonkey.

```
// @match        https://cripto.tiiny.site
// @grant        none
// ==/UserScript==
```

Figura 2: Función match usada en TamperMonkey.

2.3. Define función que obtiene automáticamente el password del documento

```
// Parte 1: Obtener la llave
const parrafo = document.querySelector("p").innerText;
const oraciones = parrafo.split(". ");
let llave = "";

for (let i = 0; i < oraciones.length; i++) {
    llave += oraciones[i].charAt(0);
}

console.log("La llave es: " + llave);
```

Figura 3: Código de Tampermonkey para obtener key de la página.

Esta función obtiene la clave usando `split()` para separar las oraciones y solo se guarda cada primer letra obtienendo así la key.

2.4. Muestra la llave por consola

Como también se muestra en la figura 3 se usa un `console.log()` para mostrar la llave por terminal.

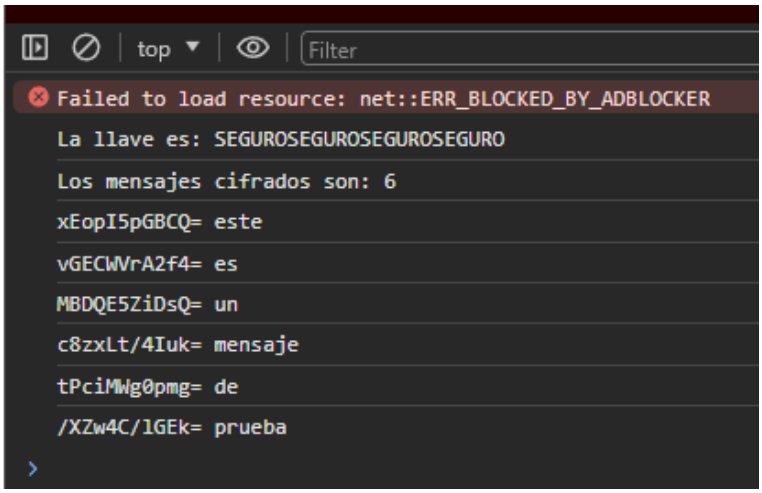

Figura 4: Salida de terminal de página web.

3. Desarrollo (Parte 2)

3.1. Reconoce automáticamente la cantidad de mensajes cifrados

Esto se logra con la cantidad de los '`div[M]`' presentes y escondidos en el navegador.

```
// Parte 2: Identificar la cantidad de mensajes cifrados
let mensajesCifrados = document.querySelectorAll('div[class^="M"]');
console.log("Los mensajes cifrados son: " + mensajesCifrados.length);
```

Figura 5: Obtención de cantidad de mensajes.

3.2. Muestra la cantidad de mensajes por consola

Siguiendo con la figura 4 tambien se muestra la cantidad de mensajes, siendo 6 mensajes en esta página.

```
<div class="M1" id="xEopI5pGBCQ=">este</div>
<div class="M2" id="vGECWVrA2f4=">es</div>
<div class="M3" id="MBDQE5ZiDsQ=">un</div>
<div class="M4" id="c8zxLt/4Iuk=">mensaje</div>
<div class="M5" id="tPciMWg0pmg=">de</div>
<div class="M6" id="/XZw4C/lGEk=">prueba</div>
</body>
```

Figura 6: Obtención de cantidad de mensajes.

4. Desarrollo (Parte 3)

4.1. Importa la librería cryptoJS

```
// Agregar la librería CryptoJS con SRI al documento HTML
var script = document.createElement('script');
script.src = 'https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js';
//script.integrity = 'sha512-INVALID_HASH_VALUE';
script.integrity = 'sha512-E8QSvWZ0eCLGk4km3hxSsNmGWbLtSCSUcewDQPQWZF6pEU8GlT8a5fF32wOl1i8ftdMhssTrF/OhyGWwonTcXA==';
script.crossOrigin = 'anonymous';
document.head.appendChild(script);
```

Figura 7: Adjuntado de CryptoJS y uso de SRI.

Se adjunta la librería de CryptoJS que será utilizada para descifrar los mensajes.

4.2. Utiliza SRI en la librería CryptoJS

Además se añade el SRI para detectar los hashes inválidos tal como se ven en la figura 9.

```
var script = document.createElement('script');
script.src = 'https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js';
script.integrity = 'sha512-INVALID_HASH_VALUE';
//script.integrity = 'sha512-E8QSvWZ0eCLGk4km3hxSsNmGWbLtSCSUcewDQPQWZF6pEU8GlT8a5fF32wO
script.crossOrigin = 'anonymous';
document.head.appendChild(script);
```

Figura 8: Código con hash inválido de SRI.

Si se modifica el hash tal como se ve en la figura 8 en la consola del navegador se muestra lo siguiente:

4.3. Repercusiones de SRI inválido

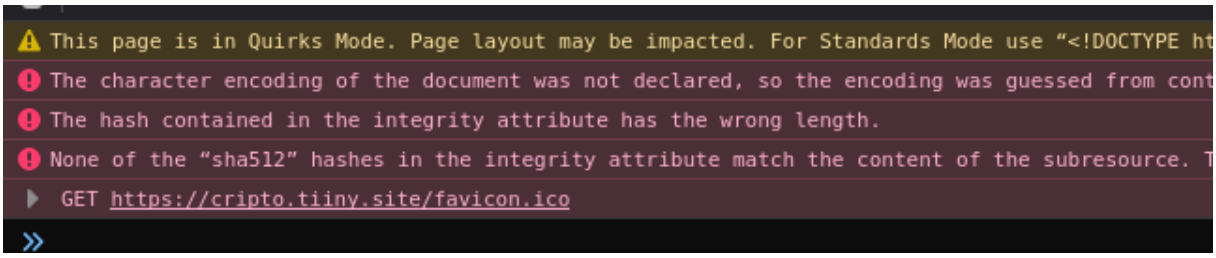

Figura 9: Detección de error con SRI.

Cuando se detecta el error de SRI el código no es válido y no se ejecuta.

4.4. Logra decifrar uno de los mensajes

Los mensajes cifrados estan en base64, por lo que hace un parse de Base64 a bytes luego usando la key obtenida del párrafo se descifra en Triple-DES los bytes obtenidos.

4.5. Imprime todos los mensajes por consola

Para descifrar e imprimir los mensajes se hizo la siguiente función de Tampermonkey.

```
// Parte 3: Obtener cada mensaje cifrado y descifrarlo
mensajesCifrados.forEach((mensajeCifrado) => {
    let mensajeCifradoBase64 = mensajeCifrado.id;
    let mensajeCifradoBytes = CryptoJS.enc.Base64.parse(mensajeCifradoBase64);
    let mensajeDescifradoBytes = CryptoJS.TripleDES.decrypt(
        { ciphertext: mensajeCifradoBytes },
        CryptoJS.enc.Utf8.parse(llave),
        { mode: CryptoJS.mode.ECB }
    );
    let mensajeDescifrado = mensajeDescifradoBytes.toString(CryptoJS.enc.Utf8);
    console.log(mensajeCifradoBase64 + " " + mensajeDescifrado);
    mensajeCifrado.innerText = mensajeDescifrado;
});
```

Figura 10: Función descifrado de mensajes.

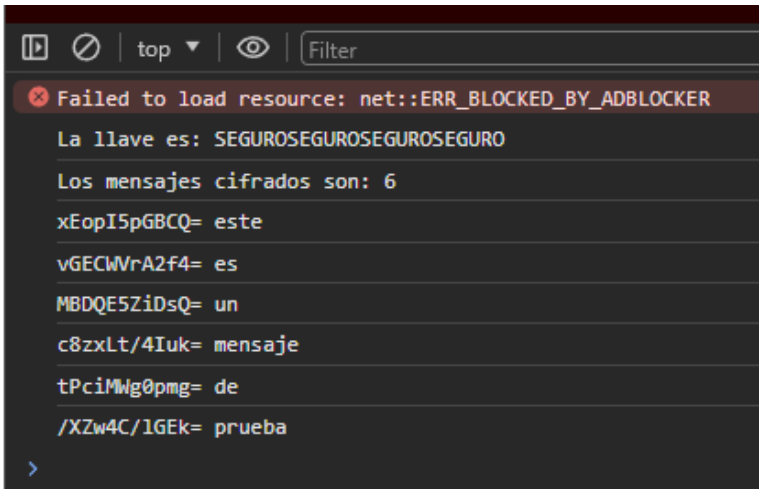

Figura 11: Output completo de mensajes en la terminal

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

Gracias también a este código se obtiene lo siguiente en la página:

evaluar, analizar y localizar las debilidades en los sistem
criptogrÃ¡ficos con el fin de encontrar debilidades en lo
criptoanÃ¡lisis, podemos comprender los criptosistemas
seguro y protegido. Resultado del criptoanÃ¡lisis es la p
responsables los criptoanalistas incluyen evaluar, analiza

este
es
un
mensaje
de
prueba

Figura 12: Mensaje obtenido mostrado en página.

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

El código funciona con otras página y esto se puede comprobar con otro link que funcione de la misma forma que el utilizado para esta experiencia.

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

Para revisar y probar este código este link le envía a la página de greasyfork correspondiente: https://greasyfork.org/es/scripts/497475-lab-cripto-cristobal-romero

Conclusiones y comentarios

En este trabajo se aplicaron programación en JS para TamperMonkey, donde se descifró un mensaje escondido en una página web de un informante.

La aplicación de funciones de CryptoJS abren un abanico de posibilidades para procedimientos de cifrado en páginas web. También se denota un caso de posibles ataques utilizando esta técnica que es necesario tener en cuenta para la seguridad de redes y sistemas HTTP.

Issues

- El adjuntar CryptoJS junto a SRI dispuso de un problema para el funcionamiento correcto, pero al conocer el uso de script.onload() permitió usar bien el sistema que se pedía en la entrega.
- Lograr reconocer que la llave para los mensajes cifrados estaba en el párrafo también supuso un problema para la realización, pero releyendo el texto se logró encontrar el patrón.
- Saber que el parse de Base64 no entregaba un mensaje de texto plano también fue un obstáculo pero buscando los usos de llaves en cifrados dió la idea de probar con obtener la llave del paragraph y probarla con los bytes.
- Para terminar con este apartado y el trabajo, encontrar que se aplicaba DES a este descifrado de los mensajes también supuso un problema, pero la reiteración y preguntandole a la asistencia de ChatGPT permitió encontrar el mensaje escondido.