Reporte de Vulnerabilidades en la Aplicación Web de Autenticación

Nombre del Estudiante: Iván Bernardo Pedrazas Rodríguez

Carrera: Ingeniería de Sistemas

Institución: Universidad del Valle

Información de Contexto
El presente informe evalúa las vulnerabilidades en el código JavaScript utilizado en la página de autenticación
en https://enlace.univalle.edu/san/webform/PAutenticar.aspx. El análisis se centró en posibles vulnerabilidades de seguridad, como inyección de código, fuga de datos sensibles y configuraciones inseguras.

Vulnerabilidades Identificadas
1. Cross-Site Scripting (XSS) - Reflejado en abrirPestana(url)

- **Ubicación**: Función abrirPestana(url)
- **Descripción**: La función abrirPestana(url) utiliza un valor url sin ninguna sanitización. Esto representa una vulnerabilidad de XSS si url toma un valor controlado por el usuario, ya que puede permitir la inyección de JavaScript malicioso.
- **PoC (Prueba de Concepto)**:
    1. Abre la consola del navegador (Ctrl+Shift+J en Chrome).
    2. Ejecuta el siguiente código en la consola para probar un posible XSS:

javascript
Copiar código
abrirPestana("javascript:alert('XSS')");

    3. Si el sitio ejecuta el alert con el mensaje "XSS", esto confirma la vulnerabilidad de XSS reflejado.
- **Impacto**: Esta vulnerabilidad puede permitir que un atacante ejecute scripts maliciosos en el navegador de un usuario, lo que podría llevar a la recolección de cookies, redireccionamiento a sitios falsos, y otras amenazas.
- **Mitigación**: Valida y escapa todas las entradas de usuario antes de procesarlas en abrirPestana y evita el uso directo de javascript: en href.

---

2. Manipulación de localStorage (Local Storage Manipulation)

- **Ubicación**: localStorage.setItem("horaServidor", addSeconds(now, 1));
- **Descripción**: La función de almacenamiento localStorage almacena el valor de horaServidor, un parámetro que influye en el comportamiento de la página.

Esto representa un riesgo si se manipula directamente, ya que un atacante podría modificar localStorage y alterar la lógica del sitio.

- **PoC**:
    1. En la consola del navegador, ejecuta:

javascript
Copiar código
localStorage.setItem("horaServidor", "manipulado");

    2. Observa si la página muestra datos incorrectos o se comporta de forma anormal.
- **Impacto**: La manipulación de valores de localStorage podría afectar la lógica del sitio o mostrar información incorrecta, lo cual podría ser aprovechado para engañar al usuario o interrumpir su experiencia.
- **Mitigación**: Evita almacenar valores críticos en localStorage y, en su lugar, utiliza valores que no puedan ser manipulados por el usuario.

---

3. Exposición de Clave API (API Key Leakage)

- **Ubicación**: var apiKey = "59548e45e0de4ca691e195137231508";
- **Descripción**: La clave de API 59548e45e0de4ca691e195137231508 es visible en el código, lo cual permite que cualquier usuario la vea y potencialmente la abuse.
- **PoC**:
    1. Utiliza la clave API para hacer una solicitud desde la consola o terminal:

bash
Copiar código
curl "https://api.weatherapi.com/v1/current.json?key=59548e45e0de4ca691e195137231508&q=Cochabamba&aqi=no"

    2. Verifica que se puede acceder a los datos meteorológicos con la clave expuesta.
- **Impacto**: Al exponer una clave API en el código, un atacante puede abusar de esta clave en otros contextos, posiblemente generando cargos adicionales y exponiendo datos del servicio.
- **Mitigación**: La clave API debe gestionarse en el servidor y no exponerse en el código JavaScript público.

---

4. Inyección de URL (Open Redirect) en abrirPestana(url)

- **Ubicación**: window.open(url, '_blank');
- **Descripción**: La función abrirPestana(url) abre cualquier URL proporcionada sin validación. Esto permite la posibilidad de redireccionamiento no autorizado si el usuario controla el valor de url.
- **PoC**:
    1. Ejecuta en la consola del navegador:

javascript
Copiar código

abrirPestana("https://evil. com");

2. La página abrirá una nueva pestaña en https://evil. com.

- **Impacto**: Esto podría llevar a un Open Redirect, donde los usuarios son redirigidos a sitios externos maliciosos, con riesgo de phishing o malware.
- **Mitigación**: Valida que la URL pertenezca al mismo dominio antes de abrirla o utiliza una lista de URLs permitidas.

---

5. Ataque de Denegación de Servicio (DoS) con Lazy-Loading en Evento scroll

- **Ubicación**: $(window).scroll(function () {...})
- **Descripción**: La función de scroll permite una carga perezosa de elementos de la página. Esto puede ser abusado si el evento scroll se desencadena rápidamente, forzando la página a realizar múltiples llamadas o procesos.
- **PoC**:
  1. En la consola del navegador, ejecuta:

javascript
Copiar código
setInterval(() => { window.scrollTo(0, document.body.scrollHeight); }, 10);

2. Observa si la página comienza a comportarse de forma lenta o si muestra errores.

- **Impacto**: Un atacante podría ralentizar la página al provocar una carga continua de contenido.
- **Mitigación**: Usa un throttle para limitar la frecuencia con la que se ejecuta la función scroll.

---

6. Prueba de Inyección de Cookies (Cookie Poisoning)

- **Ubicación**: setCookie("clima", clima, 2); setCookie("climaImagen", climaImagen, 2);
- **Descripción**: Las cookies clima y climaImagen se generan sin validación de origen, lo que permite a un atacante modificarlas para inyectar valores inseguros.
- **PoC**:
  1. En la consola, prueba modificar las cookies:

javascript
Copiar código
document.cookie = "climaImagen=https://sitio-malicioso.com/icon.png";

2. Refresca la página y verifica si el sitio carga la imagen de la URL maliciosa.

- **Impacto**: Esto permite a un atacante manipular datos sensibles, lo cual podría llevar a la modificación visual de la página o a la inserción de contenido malicioso.
- **Mitigación**: Valida y restringe los valores de cookies, y considera usar cookies HttpOnly o Secure.

---

7. Fuga de Información en la Consola

- **Ubicación**: console.log("Cargo nuevo: " + clima + ", " + climaImagen);
- **Descripción**: La consola registra información sensible sobre el estado de las cookies y otras variables internas, lo cual puede ayudar a un atacante a obtener información sobre la lógica del sitio.
- **PoC**:
    1. Abre la consola en el navegador y observa los mensajes de console.log.
    2. Observa valores internos como clima y climaImagen que aparecen en la consola.
- **Impacto**: La exposición de información en la consola facilita el análisis del sistema y la recolección de datos útiles para ataques.
- **Mitigación**: Elimina los console.log en producción o configura un mecanismo seguro de monitoreo sin exponer datos internos.

Conclusión

El análisis reveló varias vulnerabilidades en la implementación JavaScript del sistema de autenticación en https://enlace.univalle.edu/san/webform/PAutenticar.aspx. Estas vulnerabilidades podrían comprometer la seguridad de los usuarios y de la infraestructura si no se abordan adecuadamente. Se recomienda la implementación de las medidas de mitigación para cada vulnerabilidad y una auditoría continua de la seguridad del código.