

Práctica Web ZAP

Docente: Arturo Villa López

Estudiante: Gabriel Antonio González López

Repositorio: <https://github.com/gabosaurio12/ASS-PracticaWebZAP>

1. Objetivo

El objetivo de la práctica es construir una web mínima con conexión a base de datos, se usará SQLite. Mediante esta construcción se debe comprender el flujo:

Construir → Detectar → Corregir → Verificar

Se usará NodeJS para esta página, al principio se dejarán vulnerabilidades a propósito.

A continuación se usará OWASP ZAP para identificar las vulnerabilidades mediante un escáneo pasivo y activo, una vez registradas las vulnerabilidades se deberán mitigar mediante estrategias.

2. Metodología

Como se explicó antes se construyó el código con ciertas vulnerabilidades para simular una construcción clásica, en la que el programador puede omitir ciertas buenas prácticas de codificación y dar como resultado algunas vulnerabilidades.

Al momento de hacer los escaneos con OWASP ZAP se tendrán que registrar las vulnerabilidades graves por corregir (se marcan en rojo).

Hay tres vulnerabilidades que se tendrán que detectar: Inyección SQL, XSS y encabezados de seguridad. Estas vulnerabilidades una vez detectadas se tendrán que mitigar.

3. Amenazas encontradas

Alertas

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 988
ETag: W/"38c-Yubeu97XYBS8TaXKkaPSreq10Ic"
Date: Thu, 02 Oct 2025 14:41:08 GMT
Connection: keep-alive
Keep-Alive: timeout=5

<tr>
<td>4</td>
<td>Mallory</td>
<td>mallory@example.com</td>
</tr>
</table>
<p>Inicio</p>
</body>
</html>

Historial Buscar Alertas Output Escaneo Activo AJAX Spider SpiderAraña +

Alertas (7) Inyección SQL

CSP: Failure to Define Directive with No Fallback (3)
Cabecera Content Security Policy (CSP) no configurada (8)
Falta de cabecera Anti-Clickjacking (8)
El servidor divulga información mediante un campo(s) de encabezado de respuesta (8)
Falta encabezado X-Content-Type-Options (8)
Atributo de elemento HTML controlable por el usuario (XSS potencial) (3)

Confianza: Low
Parámetro: name
Ataque:
Evidencia: HTTP/1.1 500 Internal Server Error
CWE ID: 89
WASC ID: 19
Origen: Activo (40018 – Inyección SQL)
Vector de Entrada: Cadena de consulta de URL
Descripción: Inyección SQL puede ser posible.

Otra información:

Solución:

Inyección SQL

Editar Alerta

Inyección SQL

URL:	http://localhost:3000/users?name=%27						
Riesgo:	High						
Confianza:	Low						
Parámetro:	name						
Ataque:	'						
Evidencia:	HTTP/1.1 500 Internal Server Error						
CWE ID:	89						
WASC ID:	19						
Descripción:	Inyección SQL puede ser posible.						
Otra información:							
Solución:	No confíe en los datos de entrada del lado del cliente, incluso si existe una validación del lado del cliente. Como norma general, escriba la verificación de los datos en el código.						
Referencias:	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html						
Etiquetas de Alerta:	<table border="1"><thead><tr><th>Clave</th><th>Valor</th></tr></thead><tbody><tr><td>POLICY_QA_FULL</td><td></td></tr><tr><td>POLICY_DENTEST</td><td></td></tr></tbody></table>	Clave	Valor	POLICY_QA_FULL		POLICY_DENTEST	
Clave	Valor						
POLICY_QA_FULL							
POLICY_DENTEST							
Cancelar Guardar							

Falta de Encabezado Anti-MIME

Editar Alerta

Falta encabezado X-Content-Type-Options

URL: <http://localhost:3000/>

Riesgo: Low

Confianza: Medium

Parámetro: x-content-type-options

Ataque:

Evidencia:

CWE ID: 693

WASC ID: 15

Descripción:

La cabecera Anti-MIME-Sniffing X-Content-Type-Options no se ha establecido en 'nosniff'. Esto permite que las versiones anteriores de Internet Explorer y Chrome realicen

Otra información:

Este problema aún se aplica a las páginas de tipo error (401, 403, 500, etc.), ya que esas páginas a menudo se ven afectadas por problemas de inyección, en cuyo caso aún existe la

Solución:

Asegúrese de que la aplicación/servidor web establece el encabezado Content-Type adecuadamente, y que establece el encabezado X-Content-Type-Options a 'nosniff' para todas las

Referencias:

[https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941\(v=vs.85\)](https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85))

Etiquetas de Alerta:

+ - ✎

Clave	Valor
CWE-693	https://cwe.mitre.org/data/dictionary/cwe-693.html
OWASP Top 10 A05	https://owasp.org/Top10/A05

Cancelar **Guardar**

Falta de Encabezado X-Powered-By

Editar Alerta

campo(s) de encabezado de respuesta HTTP ""X-Powered-By""

URL: http://localhost:3000/

Riesgo: Low

Confianza: Medium

Parámetro:

Ataque:

Evidencia: X-Powered-By: Express

CWE ID: 497

WASC ID: 13

Descripción:
El servidor de la web/aplicación está divulgando información mediante uno o más encabezados de respuesta HTTP ""X-Powered-By"". El acceso a tal información podría facilitarle a los atacantes la identificación de otros marcos/componentes

Otra información:

Solución:
Asegúrese de que su servidor web, servidor de aplicaciones, balanceador de carga, etc. está configurado para suprimir las cabeceras "X-Powered-By".

Referencias:
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework
<https://www.trovhunt.com/2012/02/shhh-dont-let-vour-res>

Etiquetas de Alerta:

Cancelar Guardar

Falta de Encabezado Anti-click-jacking

Editar Alerta

Falta de cabecera Anti-Clickjacking

URL:

Riesgo:

Confianza:

Parámetro:

Ataque:

Evidencia:

CWE ID:

WASC ID:

Descripción:
La respuesta no protege contra ataques de "ClickJacking". Debes incluir Content-Security-Policy con la directiva "frame-ancestors" o X-Frame-Options.

Otra información:

Solución:
Los navegadores web modernos admiten las cabeceras HTTP Content-Security-Policy y X-Frame-Options. Asegúrese de que una de ellas está configurada en todas las páginas web

Referencias:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>

Etiquetas de Alerta:

Clave	Valor
OWASP_2021_A05	https://owasp.org/Top10/A0...
OWASP_2017_A05	https://owasp.org/www/proj...

CSP no configurada

Editar Alerta

Cabecera Content Security Policy (CSP) no configurada

URL: http://localhost:3000/

Riesgo: Medium

Confianza: High

Parámetro:

Ataque:

Evidencia:

CWE ID: 693

WASC ID: 15

Descripción:
La Política de seguridad de contenido (CSP) es una capa adicional de seguridad que ayuda a detectar y mitigar ciertos tipos de ataques, incluidos Cross Site Scripting (XSS) y ataques

Otra información:

Solución:
Asegúrese de que su servidor web, servidor de aplicaciones, balanceador de carga, etc. esté configurado para establecer la cabecera Content-Security-Policy.

Referencias:
https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetseries.owasp.org/cheatsheets/Content_Sec

Etiquetas de Alerta:

+ - ✎

Clave	Valor
CWE-693	https://cwe.mitre.org/data/definitions/693.html
OWASP 2021 A05	https://owasp.org/Top10/A05

Cancelar **Guardar**

Falta CSP

Editar Alerta

CSP: Directiva Wildcard

URL: `http://localhost:3000/`

Riesgo: Medium

Confianza: High

Parámetro: Content-Security-Policy

Ataque:

Evidencia: `'self' https: 'unsafe-inline';upgrade-insecure-requests`

CWE ID: 693

WASC ID: 15

Descripción:
Content Security Policy (CSP) es una capa de seguridad añadida que ayuda a detectar y mitigar ciertos tipos de ataques. Incluyendo (pero no limitado a) Cross Site Scripting (XSS), y ataques de inyección de datos. Estos ataques se utilizan para

Otra información:
Las siguientes directivas permiten fuentes comodín (o ancestros), no están definidas, o están definidas de forma demasiado amplia:
`style-src font-src`

Solución:
Asegúrese de que su servidor web, servidor de aplicación, balanceador de carga, etc. está configurado apropiadamente para establecer la cabecera de Política de Seguridad de Contenido.

Referencias:
<https://www.w3.org/TR/CSP/>
<https://caniuse.com/#search=content+security+policy>
<https://content-security-policy.com/>
<https://github.com/HtmlUnit/htmlunit-csp>

Etiquetas de Alerta:

Cancelar **Guardar**

CSP-Wildcard

Editar Alerta

CSP: Directiva Wildcard

URL: `http://localhost:3000/`

Riesgo: Medium

Confianza: High

Parámetro: Content-Security-Policy

Ataque:

Evidencia: `'self' https: 'unsafe-inline';upgrade-insecure-requests`

CWE ID: 693

WASC ID: 15

Descripción:
 Content Security Policy (CSP) es una capa de seguridad añadida que ayuda a detectar y mitigar ciertos tipos de ataques. Incluyendo (pero no limitado a) Cross Site Scripting (XSS), y ataques de inyección de datos. Estos ataques se utilizan para

Otra información:
 Las siguientes directivas permiten fuentes comodín (o ancestros), no están definidas, o están definidas de forma demasiado amplia:
`style-src font-src`

Solución:
 Asegúrese de que su servidor web, servidor de aplicación, balanceador de carga, etc. está configurado apropiadamente para establecer la cabecera de Política de Seguridad de Contenido.

Referencias:
<https://www.w3.org/TR/CSP/>
<https://caniuse.com/#search=content+security+policy>
<https://content-security-policy.com/>
<https://github.com/HtmlUnit/htmlunit-csp>

Etiquetas de Alerta:

+ - ✎

Cancelar **Guardar**

4. Código vulnerable

Nota: El código es inseguro a propósito

app.js

```

js app.js > ...
1  require('dotenv').config();
2  const path = require('path');
3  const express = require('express');
4  const db = require('./db');

5
6  const app = express();
7  app.set('view engine', 'ejs');
8  app.set('views', path.join(__dirname, 'views'));
9  app.use(express.urlencoded({ extended: true }));
10 app.use(express.json());

11
12 // Vulnerabilidad inyección SQL
13 √ app.get('/users', (req, res) => {
14     const name = req.query.name || '';
15     const sql = `SELECT id, name, email FROM users WHERE name LIKE '%${name}%'`;
16 √   db.all(sql, (err, rows) => {
17       if (err) return res.status(500).send('Error DB');
18       res.render('users', {rows, q:name });
19   });
20 });
21
22 // XSS reflejado
23 √ app.get('/echo', (req, res) => {
24     const msg = req.query.msg || ' ';
25     res.send(`<h1>Echo</h1><p>${msg}</p>`); // VULNERABLE
26 });
27
28 app.get('/', (req, res) => res.render('index'));
29
30 const port = process.env.PORT || 3000;
31 app.listen(port, () => console.log(`App en http://localhost:${port}`));

```

db.js

```

js db.js > ...
1  const sqlite3 = require('sqlite3').verbose();
2  const fs = require('fs');
3  const path = require('path');
4
5  const DB_PATH = process.env.DB_PATH || path.join(__dirname, 'app.db');
6  const db = new sqlite3.Database(DB_PATH);

7
8  db.serialize(() => {
9      db.run("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, email TEXT NOT NULL UNIQUE)");
10     db.get("SELECT COUNT(*) AS c FROM users", (err, row) => {
11         if (err) return;
12         if (row && row.c === 0) {
13             const seedSql = fs.readFileSync(path.join(__dirname, 'seed.sql'), 'utf8');
14             db.exec(seedSql);
15         }
16     });
17 });
18
19 module.exports = db;
20

```

5. Amenazas encontradas después de mitigar amenazas

Alertas

The screenshot shows the ZAP 2.16.1 interface with a session titled "Sesión sin Nombre - 20251007-180704 - ZAP 2.16.1". The top menu includes Archivo, Editar, Ver, Analizar, Informe, Herramientas, Importar, Exportar, En línea, Ayuda, and various system icons. The left sidebar shows "Contextos" and "Sitios", with "Sitios" expanded to show "http://localhost:3000". The main pane displays the "Cabecera: Vista Raw" (Header: Raw View) and "Cuerpo: Vista Raw" (Body: Raw View). The header shows standard HTTP headers like Content-Security-Policy, Cross-Origin-Opener-Policy, and Cross-Origin-Resource-Policy. The body shows an HTML response with a list of users and an echo endpoint. Below this is the "Alertas" (Alerts) tab, which lists four alerts under "Alertas (4)":

- CSP: Directiva Wildcard
- CSP: Failure to Define Directive with No Fallback (2)
- CSP: style-src unsafe-inline
- Divulgación de error de aplicación

A tooltip on the right explains the alerts tab: "Aquí se mostrará el detalle completo de cualquier alerta seleccionada. Puede añadir alertas de forma manual haciendo clic con el botón derecho sobre cualquier entrada en el historial y seleccionando 'Añadir alerta'. También puede editar las alertas existentes haciendo doble clic sobre ellas."

At the bottom, there are status indicators for "Alertas" (0), "Proxy Principal" (localhost:8080), and "Current Status" (various metrics like 0 errors, 0 warnings, etc.).

CSP Wildcard

Editar Alerta

CSP: Directiva Wildcard

URL: `http://localhost:3000/`

Riesgo: Medium

Confianza: High

Parámetro: Content-Security-Policy

Ataque:

Evidencia: `'self' https: 'unsafe-inline';upgrade-insecure-requests`

CWE ID: 693

WASC ID: 15

Descripción:
Content Security Policy (CSP) es una capa de seguridad añadida que ayuda a detectar y mitigar ciertos tipos de ataques. Incluyendo (pero no limitado a) Cross Site Scripting (XSS), y ataques de inyección de datos. Estos ataques se utilizan para

Otra información:
Las siguientes directivas permiten fuentes comodín (o ancestros), no están definidas, o están definidas de forma demasiado amplia:
`style-src font-src`

Solución:
Asegúrese de que su servidor web, servidor de aplicación, balanceador de carga, etc. está configurado apropiadamente para establecer la cabecera de Política de Seguridad de Contenido.

Referencias:
<https://www.w3.org/TR/CSP/>
<https://caniuse.com/#search=content+security+policy>
<https://content-security-policy.com/>
<https://github.com/HtmlUnit/htmlunit-csp>

Etiquetas de Alerta:

Cancelar **Guardar**

CSP Style-src unsafe

Editar Alerta

CSP: style-src unsafe-inline	<input type="button" value="▼"/>
URL: http://localhost:3000/	<input type="button" value="▼"/>
Riesgo: Medium	<input type="button" value="▼"/>
Confianza: High	<input type="button" value="▼"/>
Parámetro: Content-Security-Policy	<input type="button" value="▼"/>
Ataque:	
Evidencia: 'self' https: 'unsafe-inline';upgrade-insecure-requests	
CWE ID: 693	<input type="button" value="◇"/>
WASC ID: 15	<input type="button" value="◇"/>
Descripción:	
Content Security Policy (CSP) es una capa de seguridad añadida que ayuda a detectar y mitigar ciertos tipos de ataques. Incluyendo (pero no limitado a) Cross Site Scripting (XSS), y ataques de inyección de datos. Estos ataques se utilizan para	
Otra información:	
style-src incluye unsafe-inline.	
Solución:	
Asegúrese de que su servidor web, servidor de aplicación, balanceador de carga, etc. está configurado apropiadamente para establecer la cabecera de Política de Seguridad de Contenido.	
Referencias:	
https://www.w3.org/TR/CSP/ https://caniuse.com/#search=content+security+policy https://content-security-policy.com/ https://github.com/HtmlUnit/htmlunit-csp	
Etiquetas de Alerta:	
<input type="button" value="+"/> <input type="button" value="-"/> <input type="button" value="edit"/>	
<input type="button" value="Cancelar"/> <input type="button" value="Guardar"/>	

Nota: Ya no se pueden observar amenazas graves

6. Código con mitigación

app.js

```

JS app.js > ...
1  require('dotenv').config();
2  const path = require('path');
3  const express = require('express');
4  const db = require('./db');

5
6  const app = express();
7  app.set('view engine', 'ejs');
8  app.set('views', path.join(__dirname, 'views'));
9  const helmet = require('helmet');
10 app.unsubscribe(helmet());
11 app.use(express.urlencoded({ extended: true }));
12 app.use(express.json());

13
14 // Ruta segura con consultas parametrizadas
15 app.get('/users-safe', (req, res) => {
16     const name = req.query.name || '';
17     const sql = "SELECT id, name, email FROM users WHERE name LIKE ?";
18     db.all(sql, [`${name}%`], (err, rows) => {
19         if (err) return res.status(500).send('Error DB');
20         res.render('users', {rows, q:name });
21     });
22 });
23
24 // views/echo.ejs (usa <%? msg %>)
25 app.get('/echo', (req, res) => {
26     const msg = req.query.msg || ' ';
27     res.render('echo', { msg });
28 });
29
30 app.get('/', (req, res) => res.render('index'));
31
32 const port = process.env.PORT || 3000;
33 app.listen(port, () => console.log(`App en http://localhost:${port}`));

```

7. Resultados

Como se puede observar se mitigaron al menos seis amenazas graves detectadas. Esto se logró usando consultas parametrizadas, lo cual es una práctica altamente recomendada en cualquier lenguaje y momento para disminuir la posibilidad de inyección SQL, lo otro que se hizo fue la prevención de XSS usando /echo, lo cual usa <%= %> en EJS.

Finalmente se usaron encabezados de seguridad con Helmet (npm i helmet).