



Universidad Veracruzana

Lic. Ingeniería de Software

Facultad de estadística y economía

Reporte de práctica web + db + zap = security

Alumno:

Abraham Cano Ramírez

Experiencia educativa:

Aspectos de seguridad para el desarrollo de software

Docente:

Arturo Villa López

Enlace de repositorio:

<https://github.com/AbrahamC13/PracticaWebDbZap>

Fecha de entrega:

08/10/2025

Reporte Practica web + db + zap = security

Objetivo:

Comprender el funcionamiento de ZAP para la detección y análisis de vulnerabilidades en los sistemas web. Se busca adquirir conocimientos sobre el uso y modificación de herramientas además de ZAP, como la modificación de características de un navegador, uso de certificados de red, etc.

Objetivos específicos:

- Construcción de un sitio web con funciones básicas (vulnerable y seguro).
- Construcción de sitio con Node.js
- Analizar flujo de datos.
- Identificación de vulnerabilidades de OWASP.
- Documentación de procesos de seguridad.

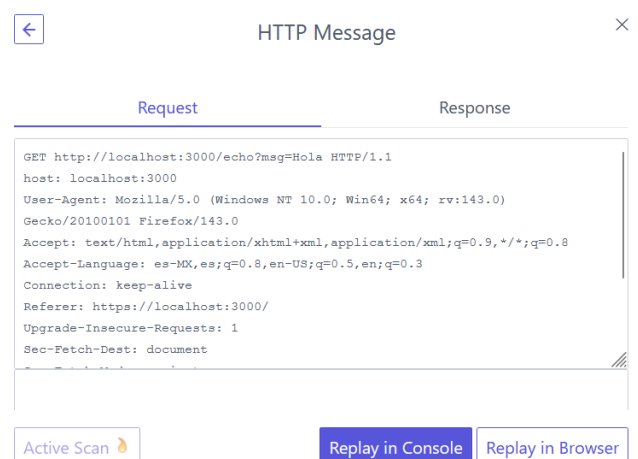
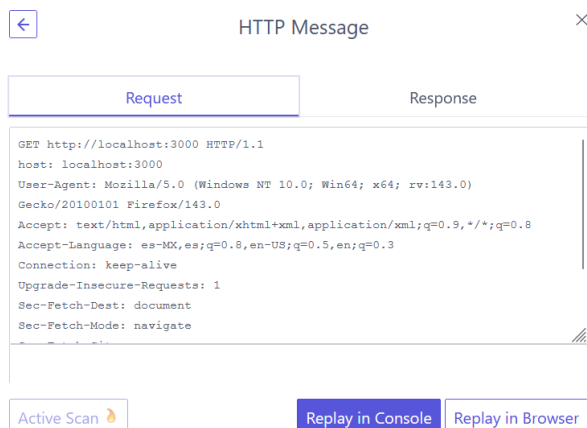
Metodología:

Herramientas: Consola, ZAP y navegador Mozilla Firefox.

1. Creación de aplicación vulnerable con Node.js
2. Instalación de ZAP
3. Configuración de navegador con el certificado de ZAP.
4. Pruebas de aplicación vulnerable con ZAP.
5. Documentación de vulnerabilidades detectadas.
6. Refactorización de aplicación con mecanismos de seguridad.
7. Pruebas de aplicación con mecanismos seguros con ZAP.
8. Documentación de las pruebas.

Hallazgos:

Capturas de los sites



←

HTTP Message

×

Request

Response

```

GET http://localhost:3000/users HTTP/1.1
host: localhost:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0)
Gecko/20100101 Firefox/143.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Referer: https://localhost:3000/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document

```

Active Scan 🔥

Replay in Console

Replay in Browser

Alertas

Alertas (4)

Cabecera Content Security Policy (CSP) no configurada (3)

GET: http://localhost:3000/

GET: http://localhost:3000/echo?msg=Hola

GET: http://localhost:3000/users

Falta de cabecera Anti-Clickjacking (3)

GET: http://localhost:3000/

GET: http://localhost:3000/echo?msg=Hola

GET: http://localhost:3000/users

El servidor divulga información mediante un campo(s) de enca

GET: http://localhost:3000/

GET: http://localhost:3000/echo?msg=Hola

GET: http://localhost:3000/users

Falta encabezado X-Content-Type-Options (3)

GET: http://localhost:3000/

GET: http://localhost:3000/echo?msg=Hola

GET: http://localhost:3000/users

Escaneo

Cabecera Content Security Policy (CSP) no configurada		Falta de cabecera Anti-Clickjacking	
URL:	http://localhost:3000/	URL:	http://localhost:3000/
Riesgo:	🔥 Medium	Riesgo:	🔥 Medium
Confianza:	High	Confianza:	Medium
Parámetro:		Parámetro:	x-frame-options
Ataque:		Ataque:	
Evidencia:		Evidencia:	
CWE ID:	693	CWE ID:	1021
WASC ID:	15	WASC ID:	15
Origen:	Pasivo (10038 - Cabecera Content Security Policy (CSP) no configurada)	Origen:	Pasivo (10020 - Cabecera Anti-Clickjacking)
Referencia de Alerta:	10038-1	Referencia de Alerta:	10020-1
Vector de Entrada:		Vector de Entrada:	

Request/response

ID	Fuente	Petición (Tiempo)	Método	URL	Código	Razón	RTT	Respuesta (Tamaño del cuerpo)	Alerta mayor	Nota	Etiquetas
1	Proxy	02/10/25, 16:47:12	GET	http://localhost:3000/	200	OK	7milisegundos	332bytes	🔥 Medio		
3	Proxy	02/10/25, 16:47:33	GET	http://localhost:3000/users	200	OK	9milisegundos	947bytes	🔥 Medio		Form
4	Proxy	02/10/25, 16:47:34	GET	http://localhost:3000/	200	OK	4milisegundos	332bytes	🔥 Medio		
5	Proxy	02/10/25, 16:47:35	GET	http://localhost:3000/echo?msg=Hola	200	OK	1milisegundos	24bytes	🔥 Medio		
6	Proxy	02/10/25, 16:47:36	GET	http://localhost:3000/	200	OK	0milisegundos	332bytes	🔥 Medio		

Las principales vulnerabilidades detectadas fueron:

Inyección SQL: Aplicación construía consultas SQL concatenando el valor del parámetro name. Permitiendo que alguien escriba código malicioso y pueda ver todos los usuarios, aunque no tenga permiso.

XSS (Cross-Site Scripting): En la página de echo, lo escrito se podía mostrar en la pantalla. Dando posibilidad de que el navegador pueda ejecutar la línea ingresada por el usuario como un programa.

Falta de encabezados de seguridad HTTP: La aplicación no incluía mecanismos de seguridad que esperan navegadores como (X-Content-Type-Options o Content-Security-Policy). Su ausencia, evitan el riesgo de ClickJacking.

Mitigaciones

1) Consultas parametrizadas – prevenir SQL inyección

Cambio de consultar para usar parámetros. En lugar de pegar directamente lo que escribe el usuario, la base de datos recibe en un espacio el valor ingresado por usuario.

```
// Ruta segura con consultas parametrizadas
app.get('/users-safe', (req, res) => {
  const name = req.query.name || '';
  const sql = "SELECT id, name, email FROM users WHERE name LIKE ?";
  db.all(sql, [`%${name}%`], (err, rows) => {
    if (err) return res.status(500).send('Error DB');
    res.render('users', { rows, q: name });
  });
});
```

2) Prevención de XSS

- Creación de archivo Echo

En vez de usar un .send y mandar lo ingresado por el usuario al HTML, se usa un .render que sólo muestra plantillas con datos, sin tomarlas como código.

```
// views/echo.ejs (usa <%= msg %>)
app.get('/echo', (req, res) => {
  const msg = req.query.msg || '';
  res.render('echo', { msg });
});
```

3) Uso de encabezados de seguridad con Helmet

- Se requirió la instalación de: npm install helmet

Instalación de Healmet, que agrega varias cabeceras de seguridad automáticamente, entre ellas, la manifestación de ventanas de terceros (ClickJacking)

```

require('dotenv').config();
const path = require('path');
const express = require('express');
const db = require('./db');
const helmet = require('helmet');

const app = express();
app.use(helmet());
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

```

Resultados del re-escaneo

Alertas (3)

- CSP: Directiva Wildcard (2)**
 - GET: http://localhost:3000/
 - GET: http://localhost:3000/echo?msg=Hola
- CSP: Failure to Define Directive with No Fallback**
 - GET: http://localhost:3000/users
- CSP: style-src unsafe-inline (2)**
 - GET: http://localhost:3000/
 - GET: http://localhost:3000/echo?msg=Hola

CSP: Directiva Wildcard

URL: http://localhost:3000/

Riesgo: Medium

Confianza: High

Parámetro: Content-Security-Policy

Ataque: default-src 'self'; base-uri 'self'; font-src 'self' https; data-form-action 'self'; frame-ancestors 'self'; img-src 'self' data; object-src 'none'; script-src 'self'; script-src-attr 'none'; style-src 'self' https; unsafe-inline; upgrade-insecure-requests

Evidencia: CWE ID: 693

WASC ID: 15

Origen: Pasivo (10055 - CSP)

Referencia de Alerta: 10055-4

Vector de Entrada:

HTTP Message

Request

Response

```

GET http://localhost:3000/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0)
Gecko/20100101 Firefox/143.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate

```

Active Scan

Replay in Console

Replay in Browser

HTTP Message

Request

Response

```

GET http://localhost:3000/echo?msg=Hola HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:143.0)
Gecko/20100101 Firefox/143.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Referer: https://localhost:3000/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document

```

Active Scan

Replay in Console

Replay in Browser

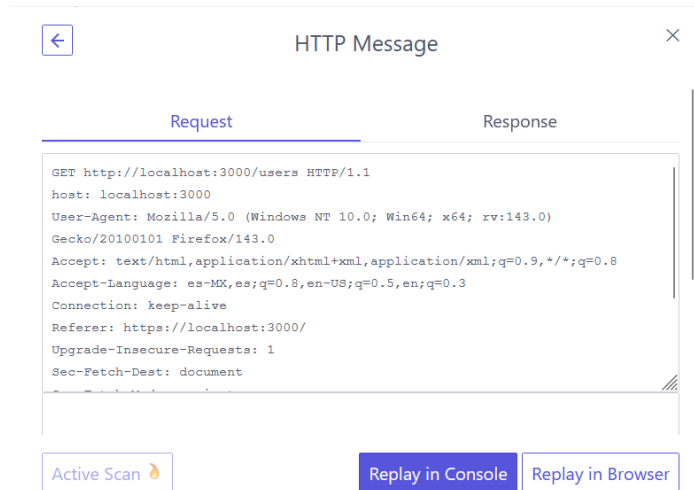


Tabla comparativa

	Antes de mecanismos de seguridad	Después de mecanismos de seguridad
Inyección SQL	Daba posibilidad de construir consultas con lo que ingresaba el usuario.	Uso de consultas parametrizadas.
XSS reflejado	Posible ejecución de códigos ingresados en el navegador.	Cambio a plantillas seguras con <code>res.render</code> y <code><% =msg %></code> . Navegador ahora solo muestra texto sin ejecutar código.
Falta de encabezados de seguridad	Posibilidad de que otra página cargue en la web un frame.	Se añadió Helmet, incluyendo encabezados de seguridad que protegen página contra ataques comunes.