PRACTICA WEB + DB + ZAP = SECURITY

ASPECTOS DE SEGURIDAD PARA EL DESARROLLO DE SOFTWARE

Arturo Villa López

Arturo Agustín Cuevas Pérez

06 de septiembre de 2025

Objetivo:

El objetivo de esta práctica fue aplicar el ciclo de vida de la seguridad en el desarrollo de software (construir → detectar → corregir → verificar) en un entorno controlado. Para esto, se construyó una aplicación web sencilla (con Node.js + Express + SQLite) con vulnerabilidades deliberadas para comprender, detectar y mitigar fallos de seguridad comunes, utilizando la herramienta de escaneo OWASP ZAP para el análisis.

Metodología y herramientas:

Esta práctica se dividió en 4 fases principales, pero antes de mencionarlas hay que indicar que tecnologías y herramientas se utilizaron

- Tecnologías: Node.js, Express.js y SQLite
- Herramienta de Análisis: OWASP ZAP

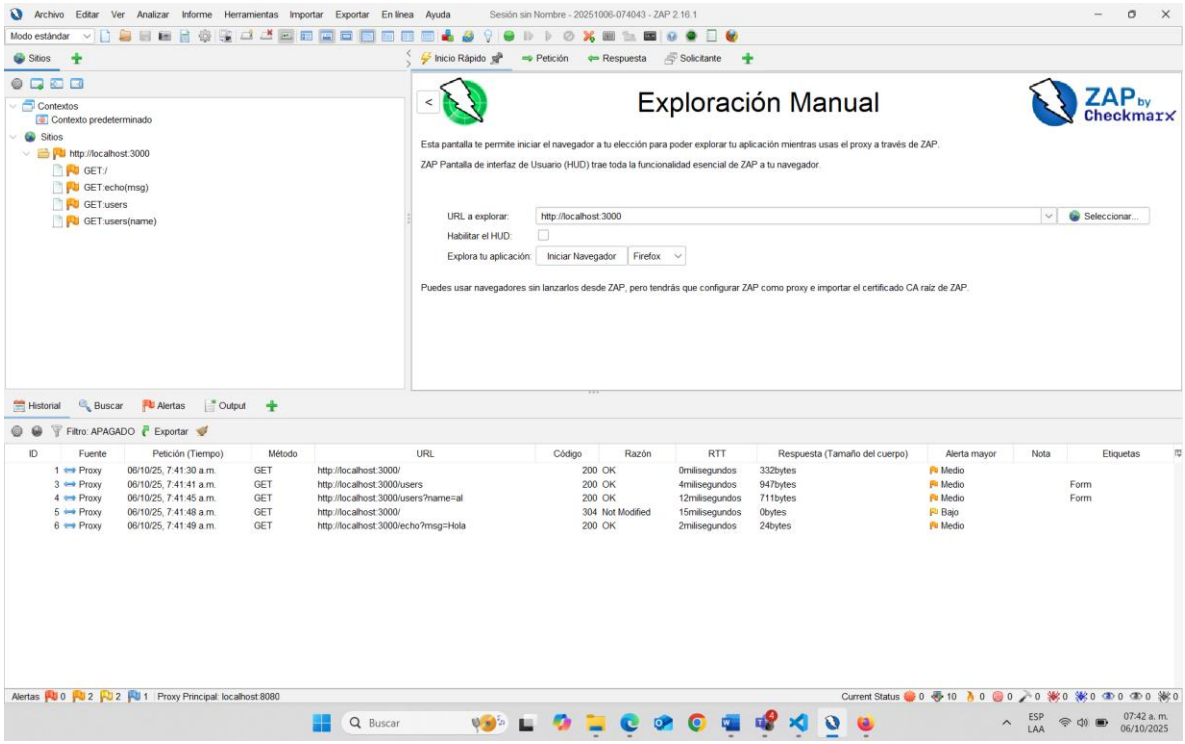
Fases de la actividad:

- Construcción de la Aplicación Vulnerable: Para comenzar se desarrolló la aplicación web (siguiendo las instrucciones marcadas en la actividad) que incluía funcionalidades básicas como un listado de usuarios desde una base de datos y un servicio de "eco". Además, con la finalidad de mitigarlas en un futuro, se introdujeron intencionadamente dos vulnerabilidades, una inyección SQL y Cross-Site Scripting (XSS).
- Escaneo y Detección: Se utilizó OWASP ZAP para realizar un escaneo pasivo y un escaneo activo sobre la aplicación ya en ejecución, todo esto con el objetivo de identificar y documentar las vulnerabilidades existentes.
- Mitigación y Refactorización del Código: Una vez identificadas las vulnerabilidades de la aplicación, se corrigió el código para eliminar dichos problemas detectados, para esto se aplicaron los cambios que se indicaban en las instrucciones de la práctica.
- Verificación y Re-escaneo: Por último, una vez ya corregidas las vulnerabilidades, se ejecutó un segundo escaneo activo con ZAP para verificar la efectividad de las mitigaciones y comparar los resultados.

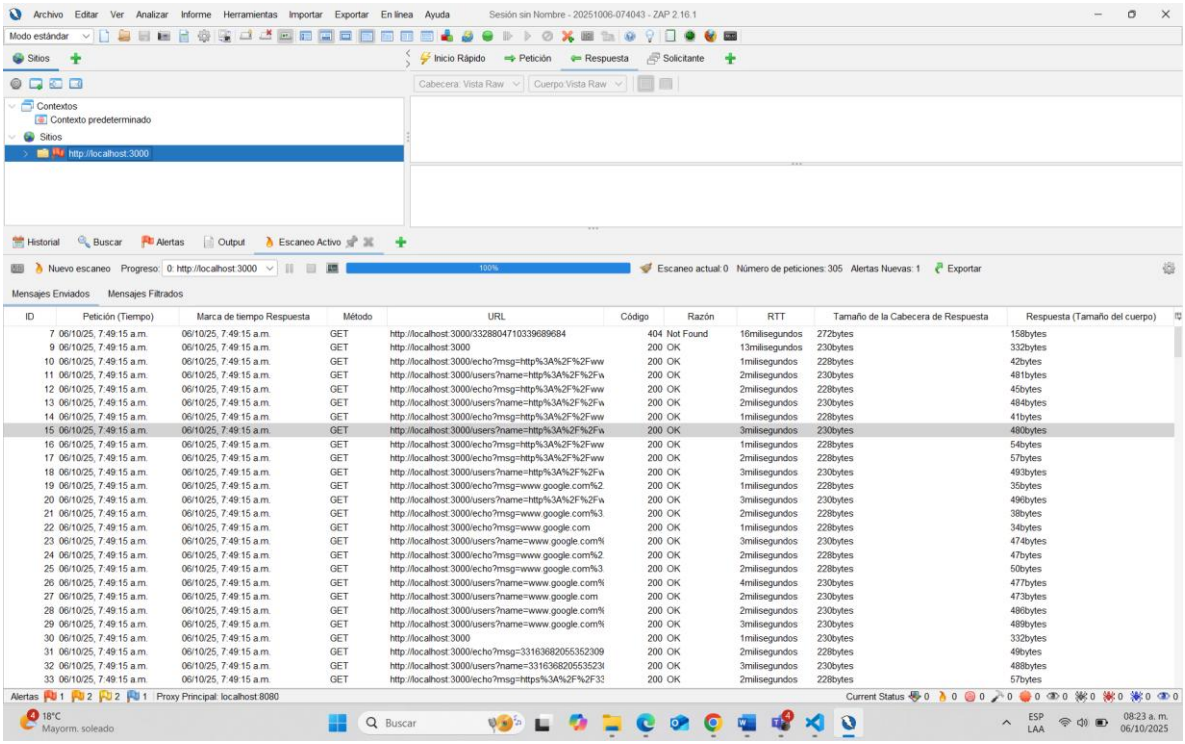
Hallazgos:

El escaneo inicial con ZAP reveló un total de 6 alertas, donde se incluía una con un riesgo alto, que comprometía la integridad de la base de datos. Los hallazgos más importantes fueron:

Pestaña **Sites** captura de endpoints:

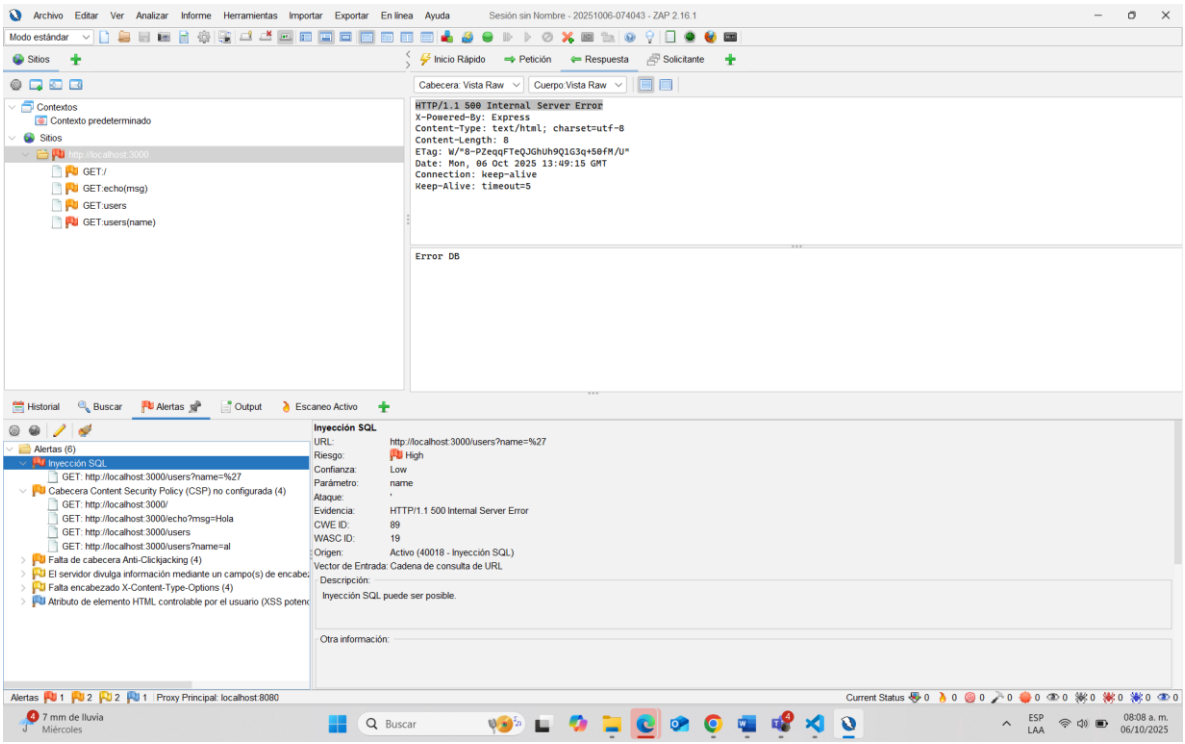


Ejecución del **Escaneo Activo:**

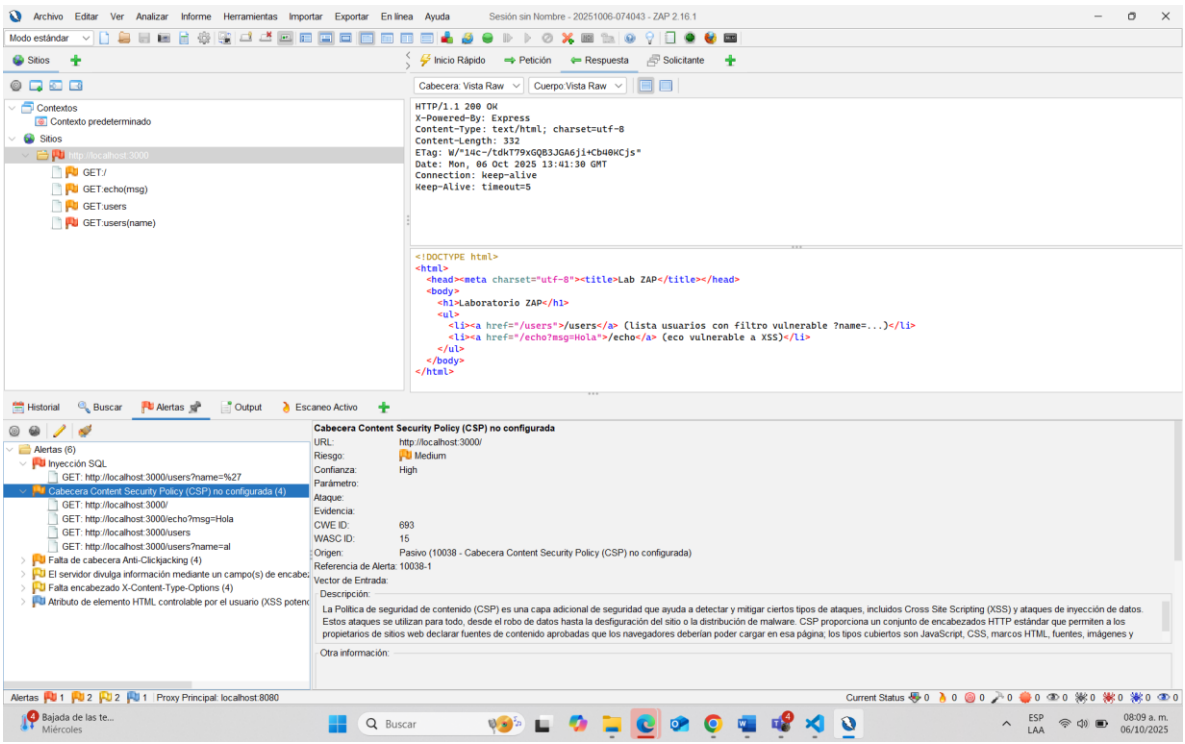


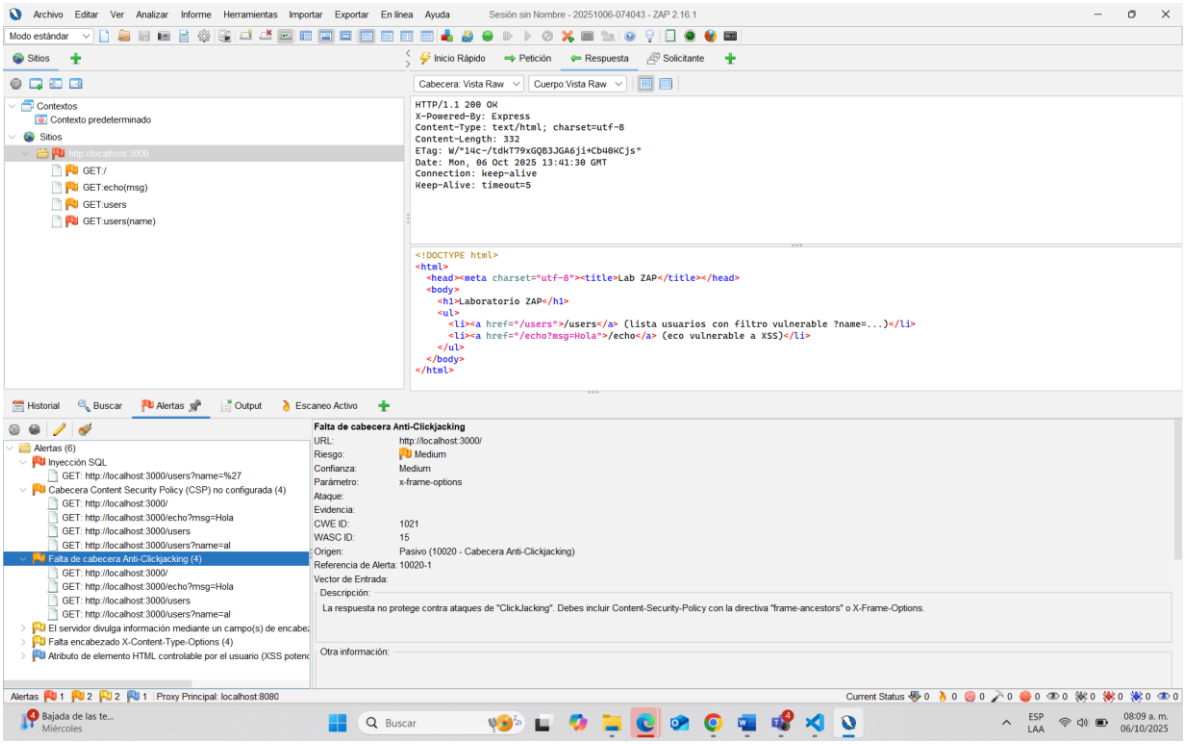
Evidencia de las **alertas** encontradas:

1. **Inyecciónn SQL:** Se detectó que el parámetro “name” en la ruta /users era vulnerable, además de que un posible atacante podía manipular la consulta a la base de datos para extraer, modificar o eliminar información.



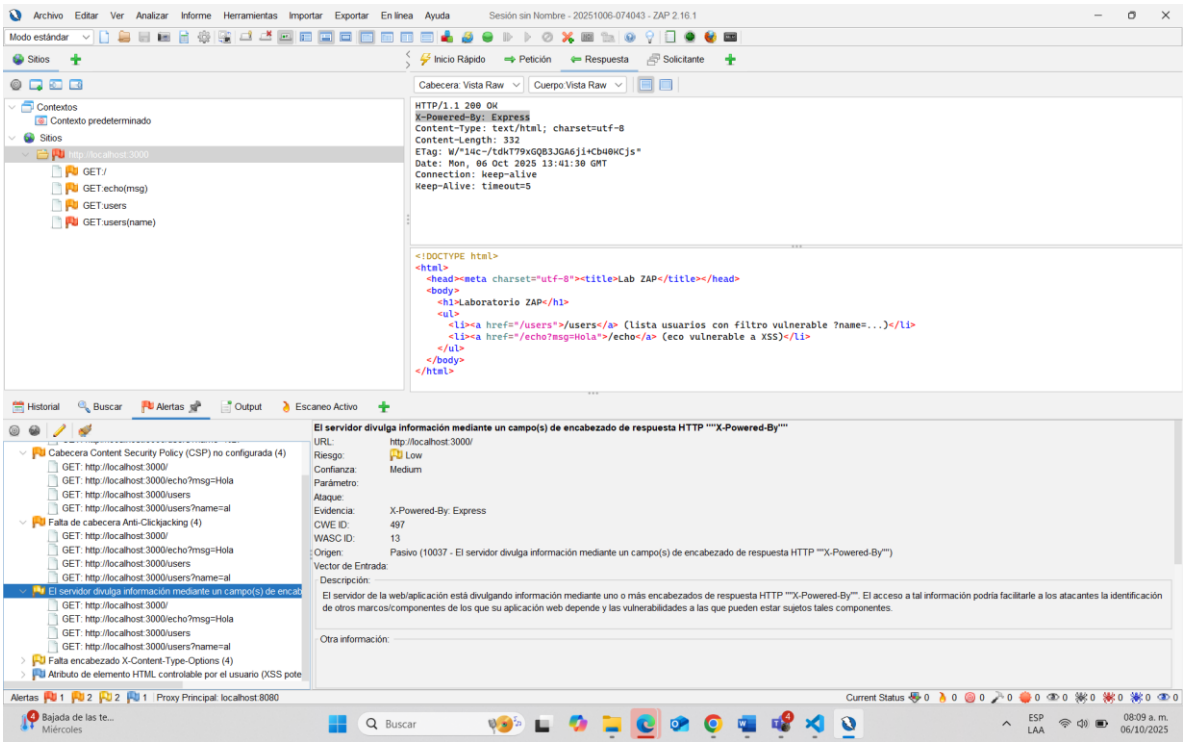
2. **Cabeceras de Seguridad Faltantes:** El análisis con ZAP demostró que había una ausencia de múltiples cabeceras HTTP que son fundamentales para la seguridad. Las detectadas fueron (Se adjuntan las capturas de pantalla de las alertas en el orden que se menciona):
    a. Content-Security-Policy (CSP): Se tenía una ausencia total de una política de seguridad de contenido.
    b. Anti-Clickjacking Header: La aplicación era vulnerable a ataques de Clickjacking.
    c. X-Powered-By: Se divulgaba información sensible sobre la tecnología del servidor.




Archivo Editar Ver Analizar Informe Herramientas Importar Exportar En línea Ayuda
Sesión sin Nombre - 20251006-074043 - ZAP 2.16.1
Modo estándar
Sitios
Inicio Rápido
Petición
Respuesta
Solicitante
Cabecera: Vista Raw
Cuerpo:Vista Raw
Contextos
Contexto predeterminado
Sitios
http://localhost:3000
GET:/
GET:echo(msg)
GET:users
GET:users(name)
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 332
ETag: W/"14c-/tdkT79xGQB3JGA6ji+Cb0WCjs"
Date: Mon, 06 Oct 2025 13:41:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"><title>Lab ZAP</title></head>
<body>
<h1>Laboratorio ZAP</h1>

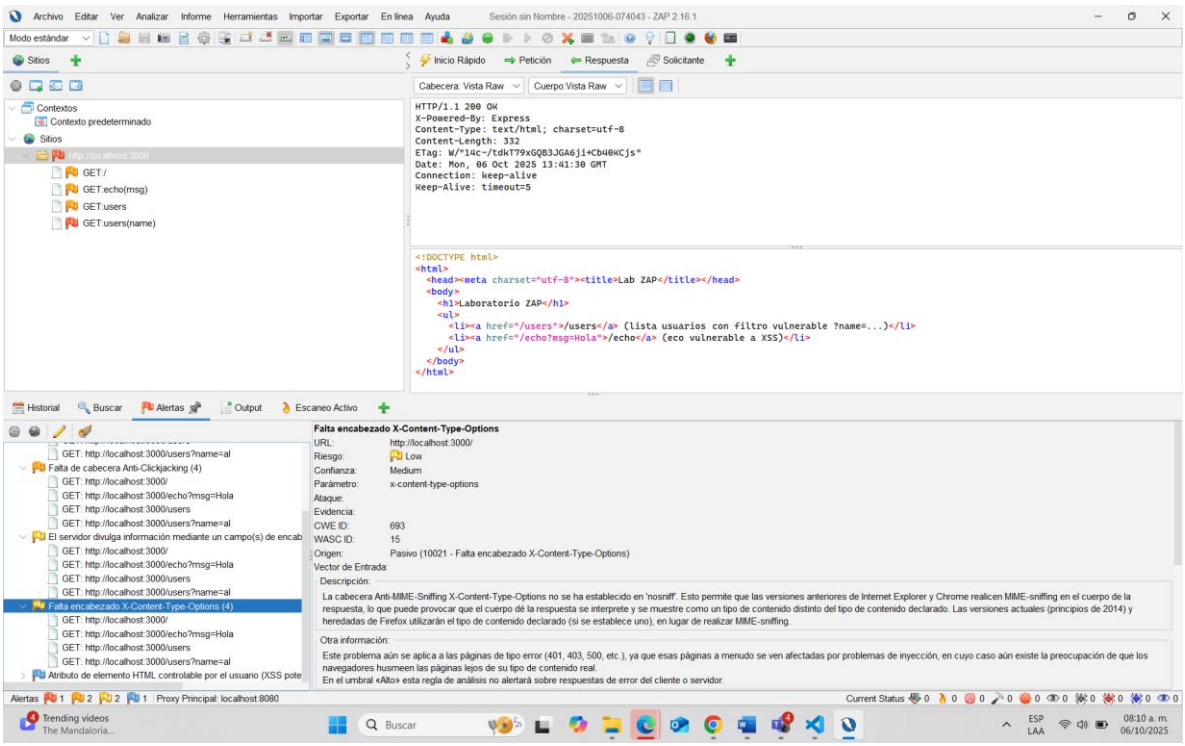
/users (lista usuarios con filtro vulnerable ?name=...)
/echo (eco vulnerable a XSS)

</body>
</html>
Historial
Buscar
Alertas
Output
Escaneo Activo
Alertas (6)
Inyección SQL
GET: http://localhost:3000/users?name=%27
Cabecera Content Security Policy (CSP) no configurada (4)
GET: http://localhost:3000/
GET: http://localhost:3000/echo?msg=Hola
GET: http://localhost:3000/users
GET: http://localhost:3000/users?name=al
Falta de cabecera Anti-Clickjacking (4)
GET: http://localhost:3000/
GET: http://localhost:3000/echo?msg=Hola
GET: http://localhost:3000/users
GET: http://localhost:3000/users?name=al
El servidor divulga información mediante un campo(s) de encabez
Falta encabezado X-Content-Type-Options (4)
Atributo de elemento HTML controlable por el usuario (XSS poten
Falta de cabecera Anti-Clickjacking
URL: http://localhost:3000/
Riesgo: Medium
Confianza: Medium
Parámetro: x-frame-options
Ataque:
Evidencia:
CWE ID: 1021
WASC ID: 15
Origen: Pasivo (10020 - Cabecera Anti-Clickjacking)
Referencia de Alerta: 10020-1
Vector de Entrada:
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:
Alertas 1 2 2 1
Proxy Principal: localhost:8080
Current Status
Bajada de las te...
Miércoles
Buscar
ESP
LAA
08:09 a. m.
06/10/2025

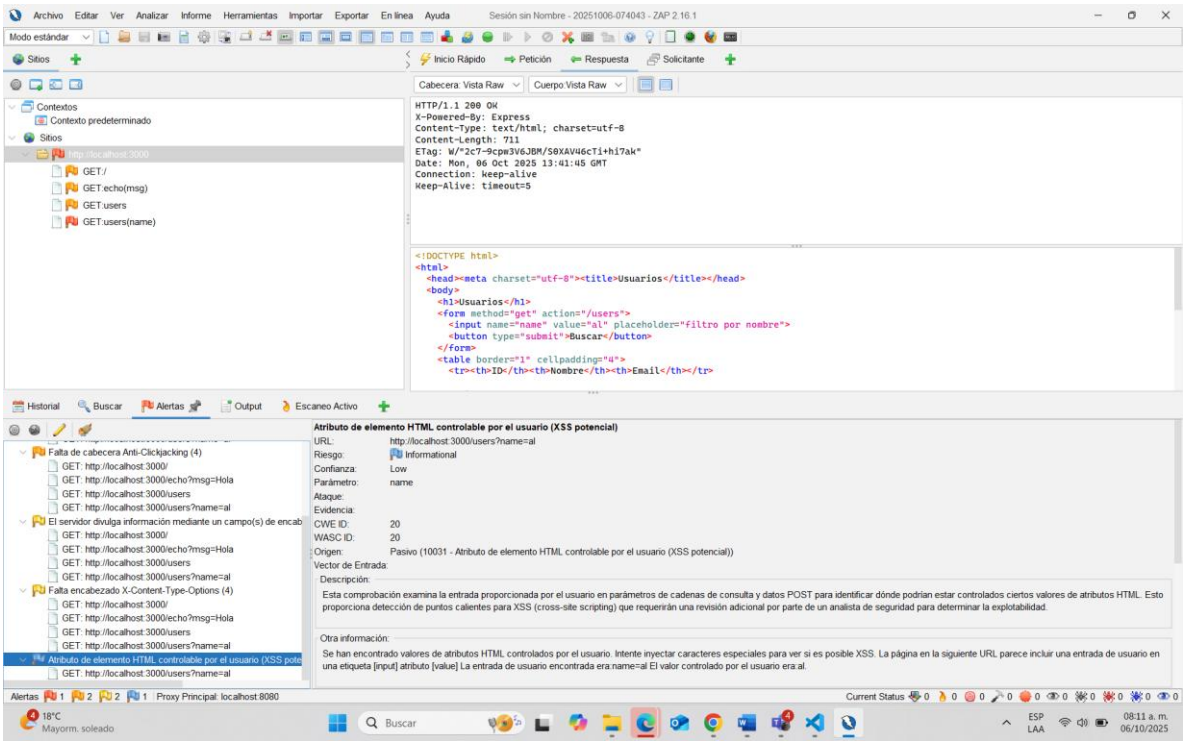

Archivo Editar Ver Analizar Informe Herramientas Importar Exportar En línea Ayuda
Sesión sin Nombre - 20251006-074043 - ZAP 2.16.1
Modo estándar
Sitios
Inicio Rápido
Petición
Respuesta
Solicitante
Cabecera: Vista Raw
Cuerpo:Vista Raw
Contextos
Contexto predeterminado
Sitios
http://localhost:3000
GET:/
GET:echo(msg)
GET:users
GET:users(name)
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 332
ETag: W/"14c-/tdkT79xGQB3JGA6ji+Cb0WCjs"
Date: Mon, 06 Oct 2025 13:41:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"><title>Lab ZAP</title></head>
<body>
<h1>Laboratorio ZAP</h1>

/users (lista usuarios con filtro vulnerable ?name=...)
/echo (eco vulnerable a XSS)

</body>
</html>
Historial
Buscar
Alertas
Output
Escaneo Activo
Cabecera Content Security Policy (CSP) no configurada (4)
GET: http://localhost:3000/
GET: http://localhost:3000/echo?msg=Hola
GET: http://localhost:3000/users
GET: http://localhost:3000/users?name=al
Falta de cabecera Anti-Clickjacking (4)
GET: http://localhost:3000/
GET: http://localhost:3000/echo?msg=Hola
GET: http://localhost:3000/users
GET: http://localhost:3000/users?name=al
El servidor divulga información mediante un campo(s) de encab
GET: http://localhost:3000/
GET: http://localhost:3000/echo?msg=Hola
GET: http://localhost:3000/users
GET: http://localhost:3000/users?name=al
Falta encabezado X-Content-Type-Options (4)
Atributo de elemento HTML controlable por el usuario (XSS pote
El servidor divulga información mediante un 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
Origen: Pasivo (10037 - El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By"")
Vector de Entrada:
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 de los que su aplicación web depende y las vulnerabilidades a las que pueden estar sujetos tales componentes.
Otra información:
Alertas 1 2 2 1
Proxy Principal: localhost:8080
Current Status
Bajada de las te...
Miércoles
Buscar
ESP
LAA
08:09 a. m.
06/10/2025



3. **XSS Potencial:** Se identificó que la ruta "/echo" reflejaba la entrada del usuario sin ningún tipo de saneamiento o escape, lo que permitía la ejecución de scripts en el navegador de la víctima**.**



Mitigaciones aplicadas:

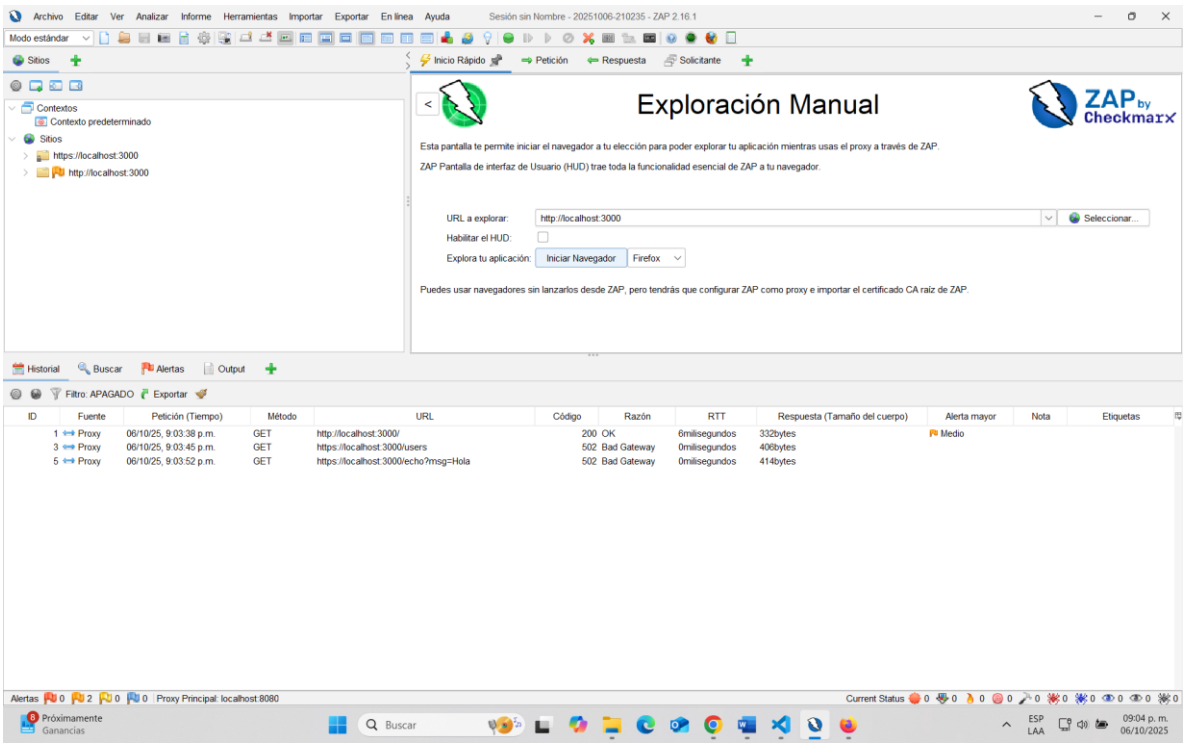
- **Prevención de Inyección SQL:** Para lograrlo se refactorizó la consulta a la base de datos para utilizar consultas parametrizadas. En lugar de concatenar la entrada del usuario directamente en el string de la consulta, se utilizó un marcador de posición (?), asegurando que la entrada siempre sea tratada como un dato y no como código ejecutable.
- **Prevención de XSS:** Se eliminó la generación de HTML directamente en app.js. En su lugar, se creó la plantilla echo.ejs y se utilizó la sintaxis "<%= msg %>", que escapa automáticamente los caracteres especiales de HTML, convirtiendo cualquier intento de inyección de script en texto plano inofensivo.
- **Implementación de cabeceras de seguridad:** Para mitigarlas se añadió "helmet" a la aplicación. Con la adicción de "app.use(helmet)" se establecieron automáticamente múltiples cabeceras de seguridad, lo que permitió que se corrigieran las alertas mencionadas con anterioridad.

Resultados del Re-escaneo y Verificación

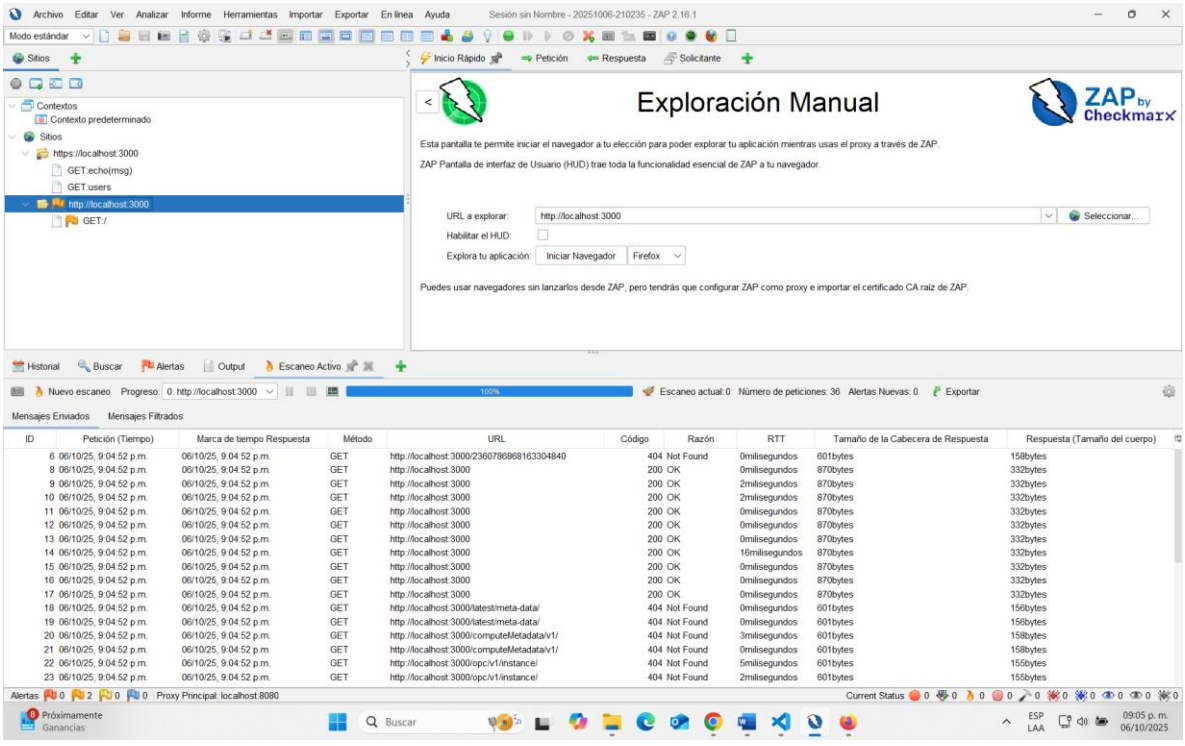
Una vez que se aplicaron los cambios para mitigar las vulnerabilidades detectadas en el primer escaneo, se realizó un segundo escaneo, este último demostró que el número de alertas pasó de 6 a 2. Además, se eliminaron el 100% de las alertas detectadas con anterioridad, incluso las alertas nuevas que fueron detectadas no fueron de un riesgo alto, sino de uno medio.

Re escaneo:

Pestana **Sites** y captura de endpoints:

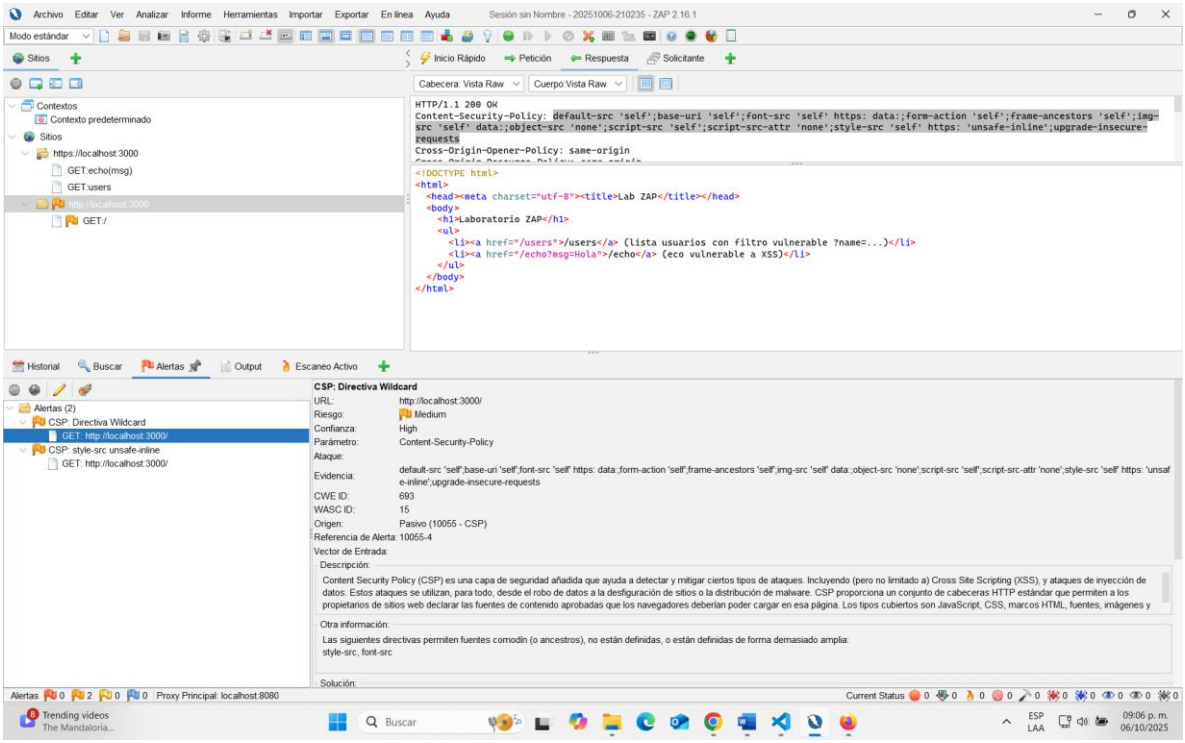


Ejecución del **Escaneo Activo:**



Evidencias de las **alertas** encontradas tras el re escaneo:

- CSP Directiva Wildcard



- CSP style-src unsafe-inline

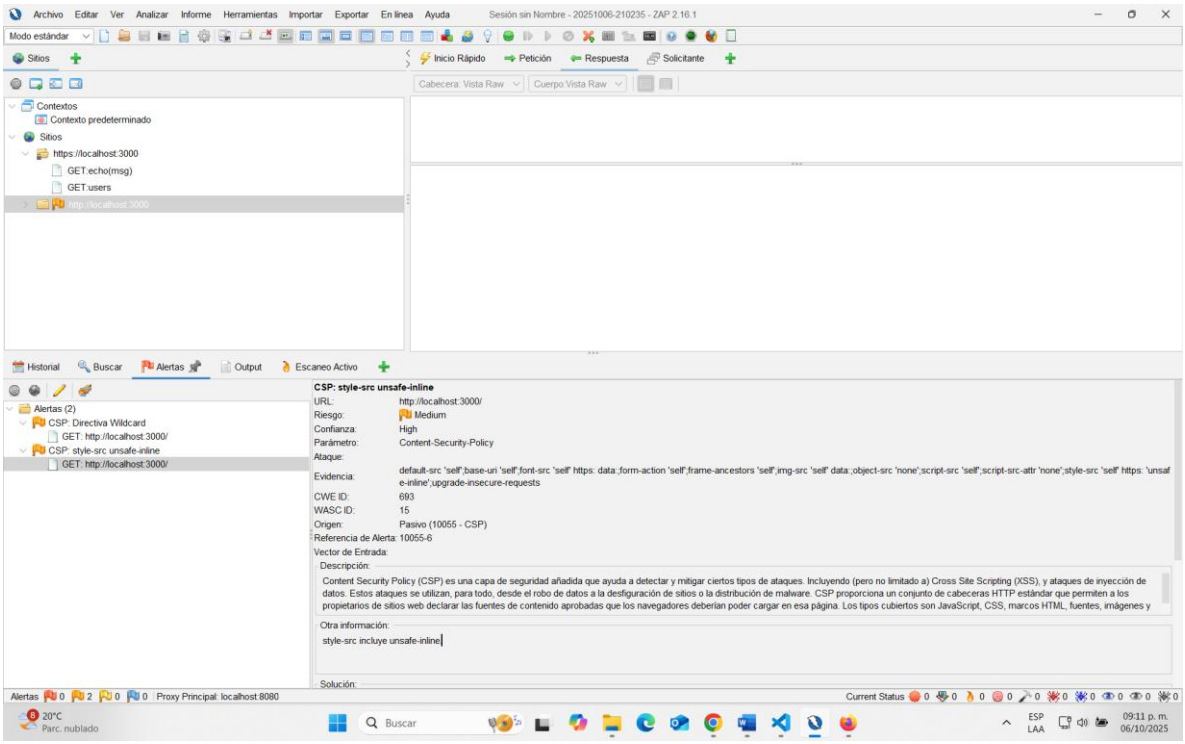


Tabla comparativa de alertas ZAP

TABLA COMPARATIVA DE ALERTAS		
Alertas detectadas en la versión vulnerable		
Vulnerabilidad / Alerta	**Análisis del Riesgo (Antes)**	**Análisis del Resultado (Después)**
Inyección SQL (*Alto*)	La aplicación era vulnerable a que un posible atacante manipulara la base de datos, debido a que la entrada del usuario en el filtro de búsqueda se concatenaba directamente en la consulta SQL, lo que permitía que se pudiera ejecutar código no deseado.	La vulnerabilidad fue eliminada por completo al refactorizar el código para utilizar consultas parametrizadas, por lo que la entrada del usuario ahora se trata como un dato y no como parte de la consulta SQL.
Multiples cabeceras de Seguridad • **CSP no configurado (*Medio*)** • **Anti-Clickjacking (*Medio*)** • **X-Content-Type-Options (*Bajo*)** • **X-Powerd-By (*Bajo*)**	Anteriormente no se incluían cabeceras HTTP esenciales, lo que provocaba que estuviéramos expuestos a riesgos como el Clickjacking, la ejecución de archivos como scripts o la divulgación de que tecnología se está usando, lo que le daría pistas a un posible atacante.	Todas estas alertas fueron solucionadas al implementar “app.use(helmet())”, esto añadió automáticamente las caberas faltantes que existían y eliminó la cabecera de X-Powered-By, lo que fortaleció la seguridad.
XSS Potencial (*Informativo*)	La entrada del usuario en la ruta /echo se reflejaba directamente en el HTML de la respuesta, lo que generaba una vulnerabilidad de XSS, donde un posible atacante podía inyectar código de JavaScript para que se ejecutara en el navegador de otra víctima.	El problema se solución al dejar de general el HTML en app.js y en su lugar se utilizó una plantilla (echo.ejs). La sintaxis “<%= msg %>” de EJS escapa automáticamente los caracteres peligrosos, lo que convertía cualquier intento de inyección de código en texto plano inofensivo
Alertas detectadas en la versión segura		

Alertas de CSP • **Directiva Wildcard (*Medio*)** • **style-src unsafe-inline (*Medio*)**	No habían sido detectadas en en código anterior porque no existía ninguna política de seguridad de contenido (CSP)	Al haber añadido “helmet”, se aplicó CSP, que a pesar de que es una mejora, ZAP indica que la política por defecto de helmet es algo permisiva, por lo que recomienda una configuración más estricta para un entorno real.