UNIVERSIDAD TECNOLÓGICA DE LA SELVA

INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

✓ ASIGNATURA:

SEGURIDAD EN EL DESARROLLO DE APLICACIONES

✓ NOMBRE DEL DOCENTE:

MTRO. ANDRÉS DONACIANO MARTÍNEZ GUILLÉN

NOMBRE DEL LOS INTEGRANTES:	*PARTICIPACIÓN*
JOSÉ ALEJANDRO DÍAZ GÓMEZ	100%
HUGO RUBÉN DÍAZ CRUZ	100%
MARIANA EURICE GUILLÉN NAVARRO	100%
FROILÁN NÁJERA MORALES	100%

✓ GRADO: 8 GRUPO:" A "TURNO: VESPERTINO

✓ LUGAR Y FECHA DE ENTREGA:

OCOSINGO, CHIAPAS A 08 DE ABRIL DEL 2025

Contenido

Introducción

En el presente trabajo se utilizó la herramienta OWASP ZAP (Zed Attack Proxy) para realizar un análisis de seguridad a una aplicación web. El objetivo principal fue identificar posibles vulnerabilidades y evaluar qué tan expuesta se encuentra la aplicación ante ataques comunes, como inyección de código, fallos en la gestión de sesiones, o configuraciones inseguras.

A través de diferentes técnicas de escaneo —como el escaneo pasivo, activo y spidering— se logró obtener un panorama general del estado de seguridad de la aplicación. Posteriormente, se analizaron las alertas generadas, enfocándose en vulnerabilidades críticas como inyección SQL o Cross Site Scripting (XSS), permitiendo así interpretar los riesgos y proponer recomendaciones para su mitigación.

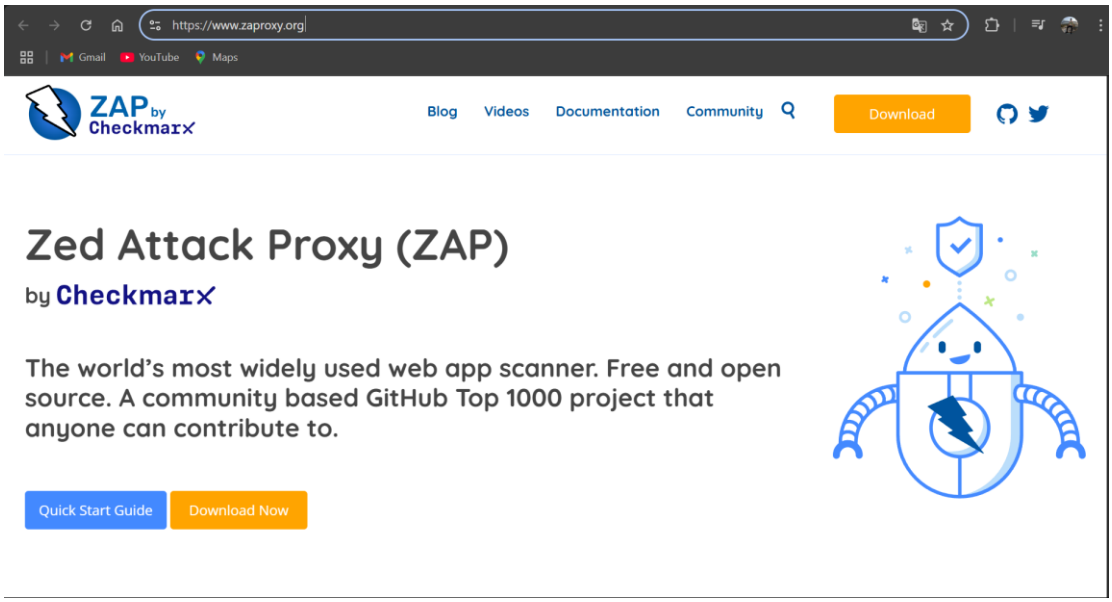
Este ejercicio forma parte del proceso de aprendizaje en pruebas de seguridad web y en el uso de herramientas automatizadas para la detección de fallos en entornos reales o simulados.

Herramienta Utilizada.

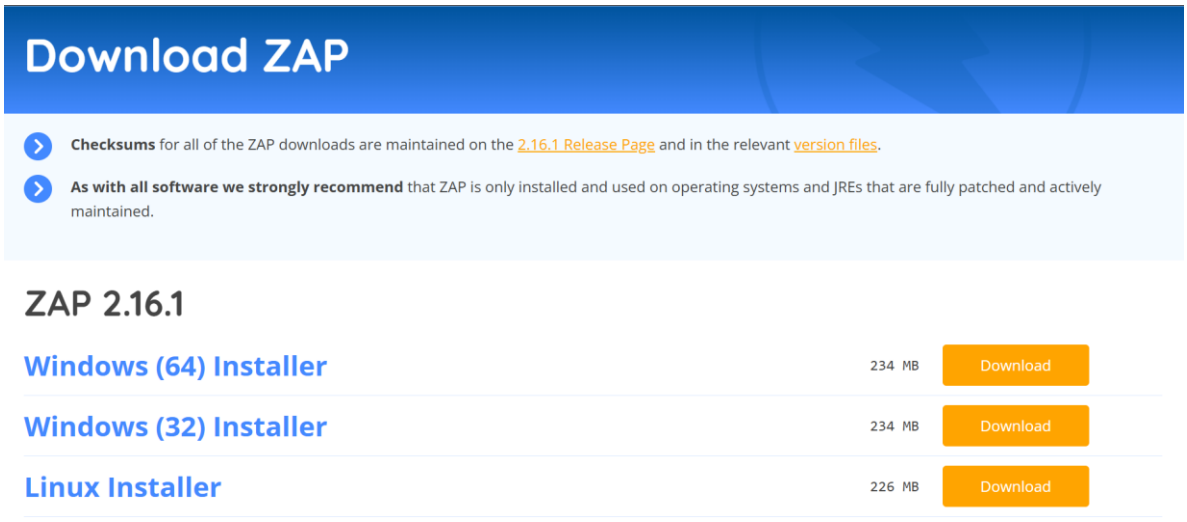
OWASP ZAP (Zed Attack Proxy)

- **Instalación.**
  Se utilizó la herramienta **OWASP ZAP (Zed Attack Proxy)**, una plataforma gratuita y de código abierto especializada en pruebas de seguridad para aplicaciones web.

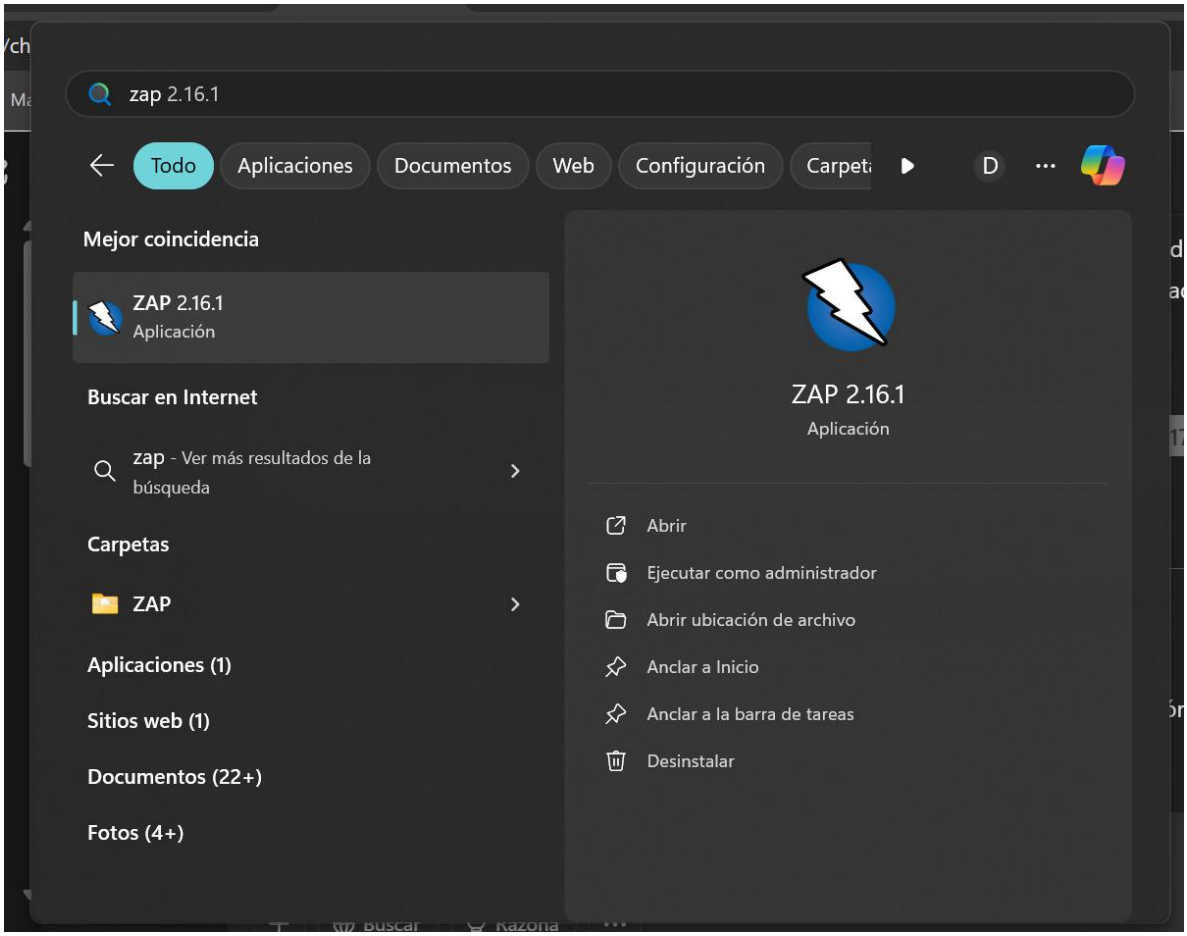
  Sitio oficial: https://www.zaproxy.org/



Seleccionamos la partición en donde queramos descargar, después que se haya descargado, la instalación solo darle next, next hasta que se finalice.



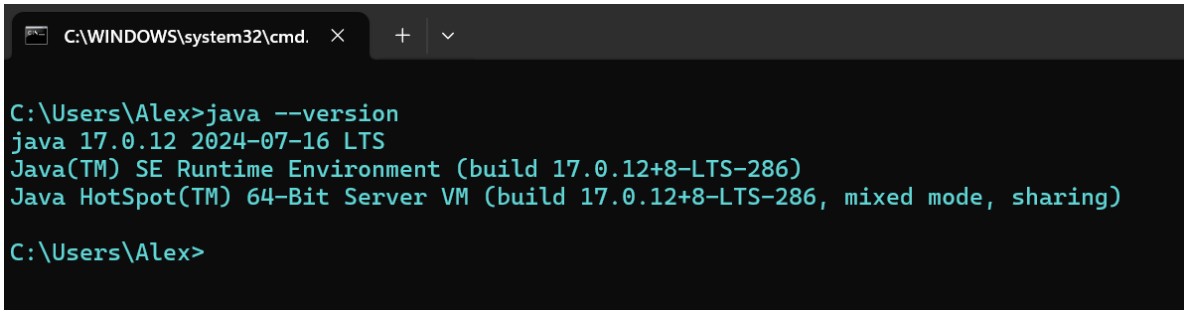
Verificamos la instalación en Windows.



Durante la instalación, fue necesario contar con Java Runtime Environment (JRE) versión 17 o superior. Se instaló Java JDK 17, lo cual permitió ejecutar el instalador de ZAP sin inconvenientes.

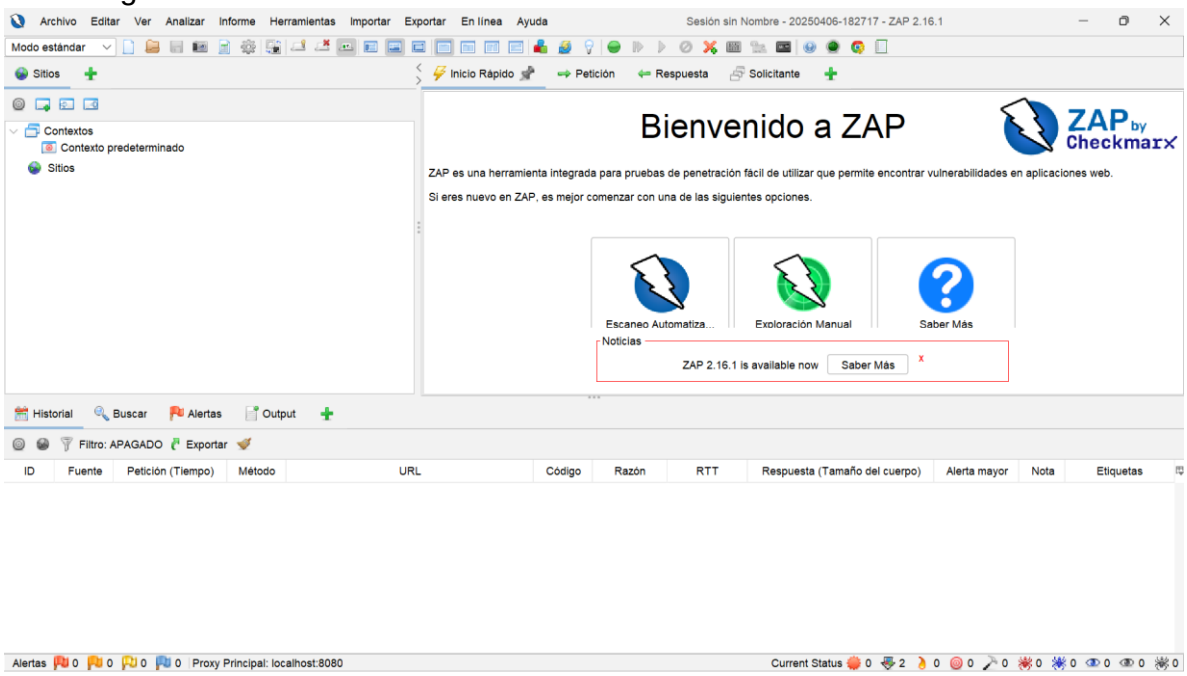
Sitio oficial: https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

Verificar la instalación de java, abrimos la terminal y escribimos java –version.



- **Pruebas empleando la herramienta.**

Interfaz gráfica de la herramienta

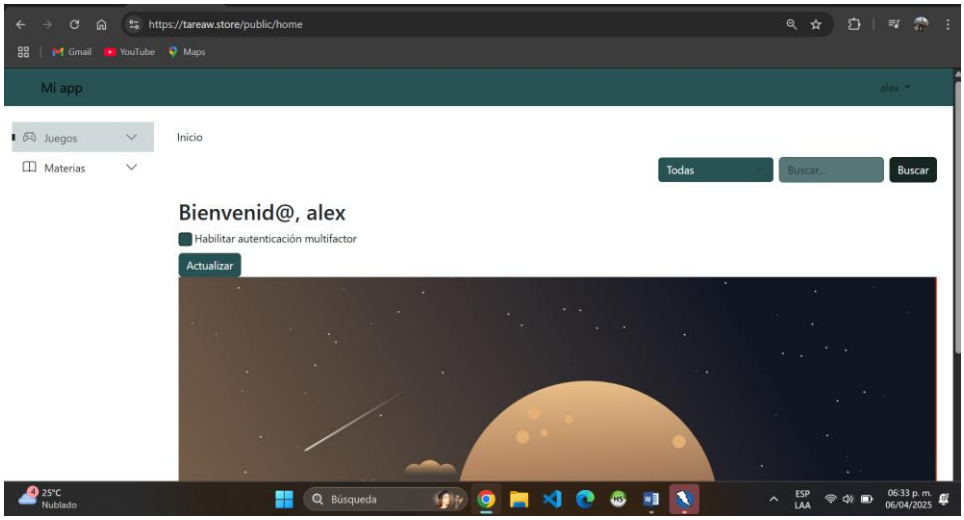


Tiene dos opciones de escaneo uno que es automática que es lo que se va a utilizar y el otro que es manual.

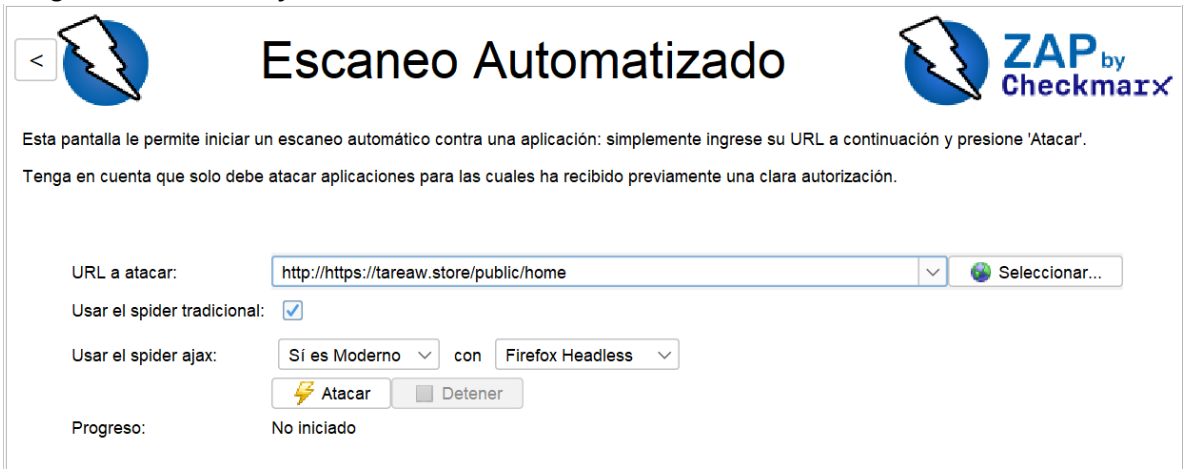
- **Identificación del tipo de pruebas que se realizó la herramienta.**

Para empezar el escaneo de la aplicación web, le damos clic en escaneo automático, abre esta ventana en donde solicita la URL.

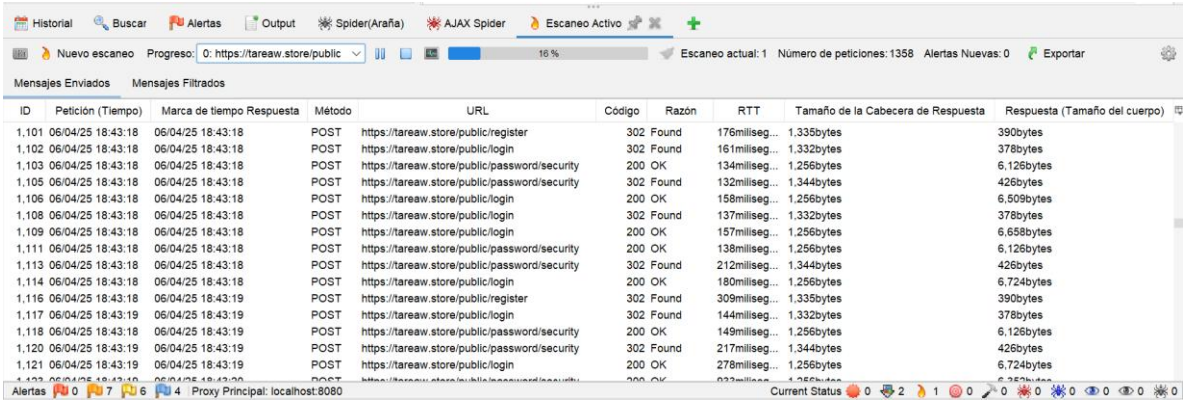
Agarramos la URL de esta aplicación alojada en un hosting que fue desarrollado por nosotros mismos para una actividad en clase.



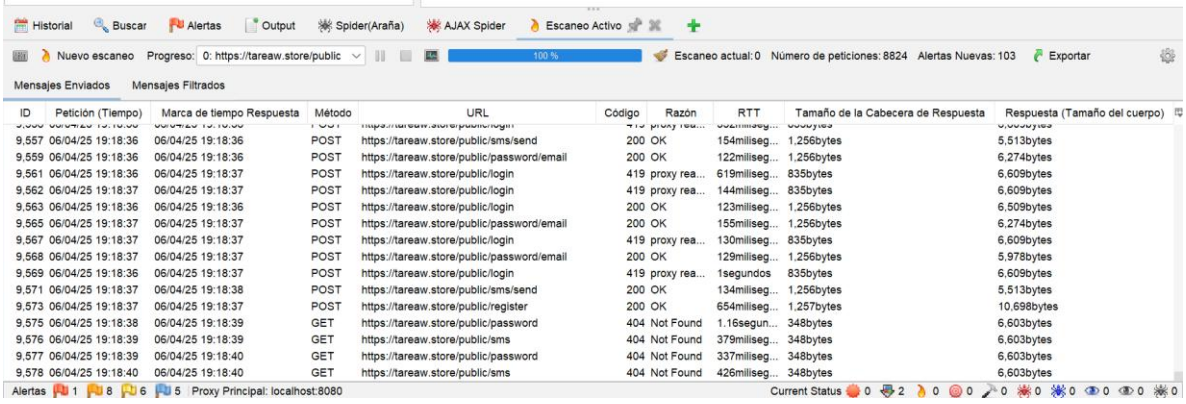
Asignamos la URL y le damos clic en atacar.



Empieza el escaneo en donde muestra opciones como la petición, método, URL, tamaño en bytes y respuestas etc..



Termina el escaneo con el ID 9,578 es decir que escaneo 9,578 veces por todo en la aplicación.



- **Identificación de los elementos que evalúa la herramienta.**

Durante el escaneo, OWASP ZAP evaluó diferentes elementos clave de la aplicación, tales como.

Formularios y campos de entrada: Para detectar inyecciones, XSS, o ausencia de validaciones.

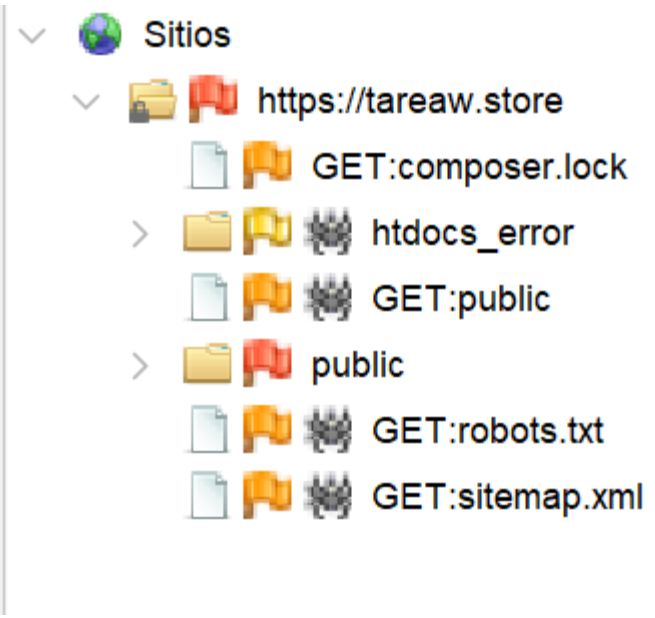
Cookies y encabezados HTTP: Para validar si están configuradas con atributos de seguridad como HttpOnly, Secure, etc.

Archivos JavaScript, HTML y recursos externos: Para detectar enlaces inseguros o vulnerabilidades en librerías.

Parámetros en URLs (GET y POST): Para encontrar puntos donde puedan inyectarse datos maliciosos.

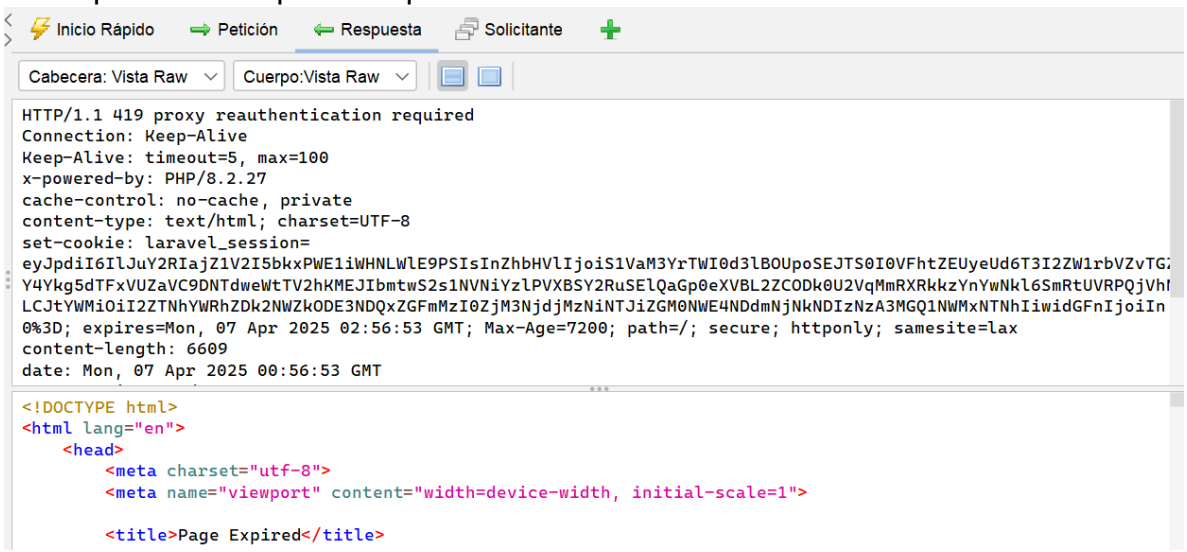
Endpoints de la aplicación: Identificando los puntos de entrada del sistema como formularios de login, registros, APIs, etc.

En las banderas rojas son alertas con alta prioridad, el color naranja son alertas con media prioridad y el amarillo son alertas con baja prioridad.

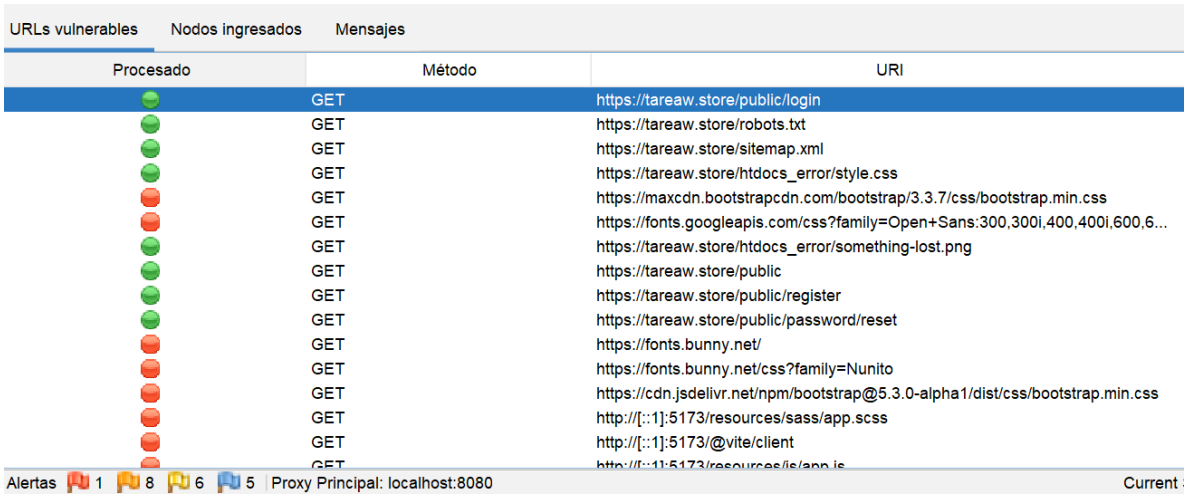


También se muestra la petición que hizo la herramienta, mostrando información como la referencia, cookie, URl entre otros.

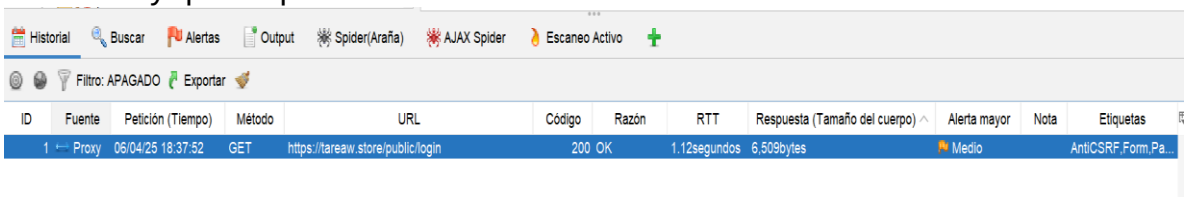
La respuesta de la petición que se hizo anteriormente.



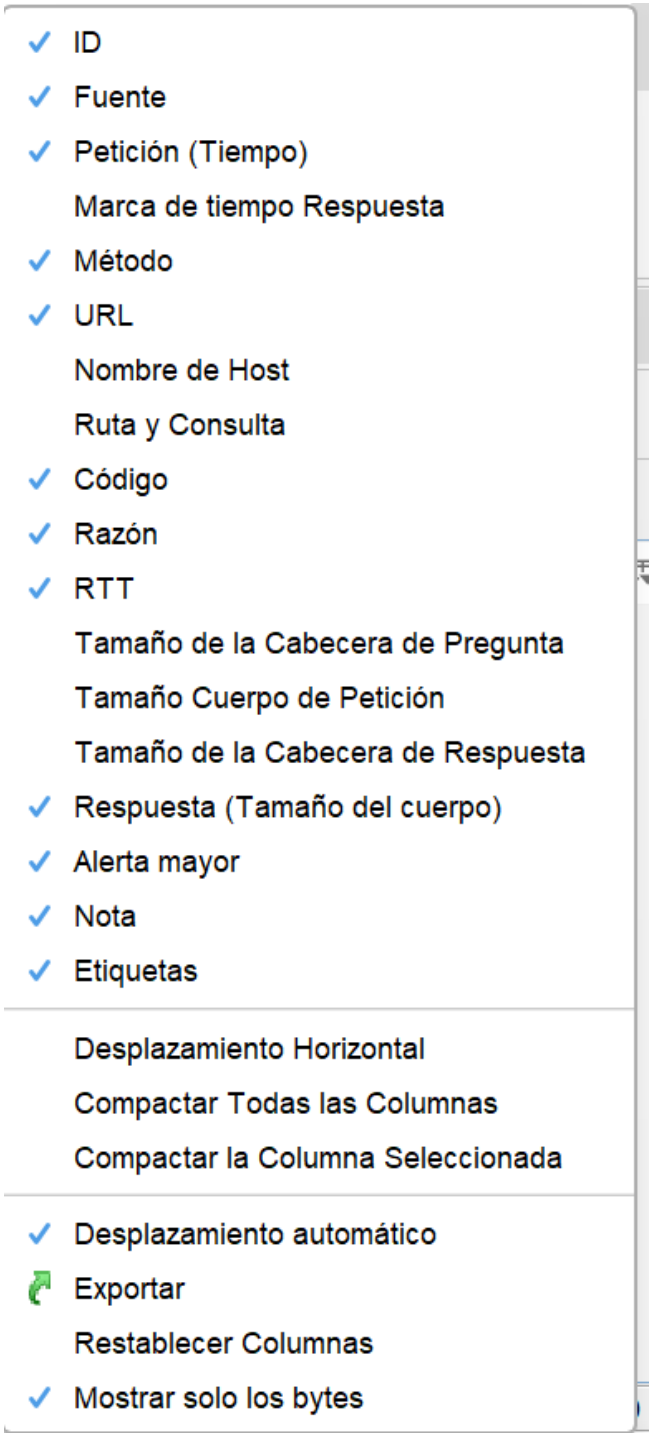
Identificación de URLs Vulnerables, mostrando el proceso y el tipo de método.



Historial completo del escaneo, es decir que muestra todas las opciones que se realizo y que se pueda visualizar en uno solo.



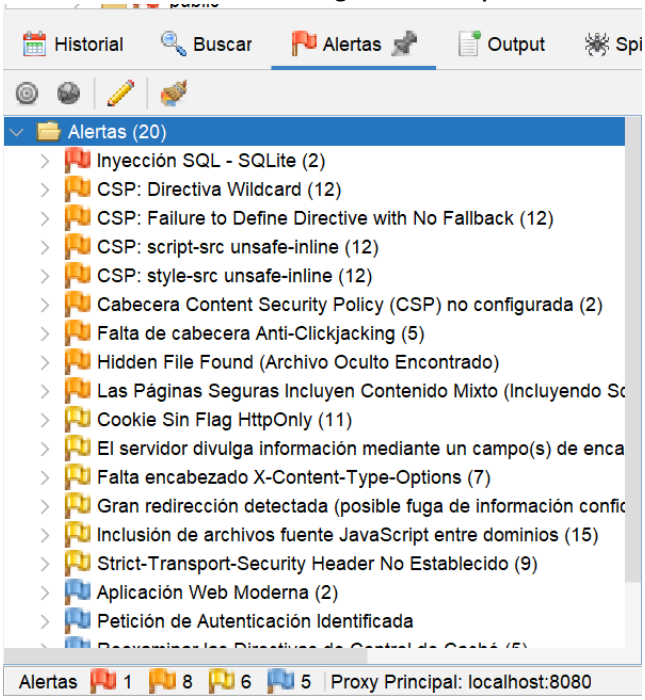
En el cuadro no se refleja todo, pero esto sería todas las opciones que se podrá visualizar en el historial.



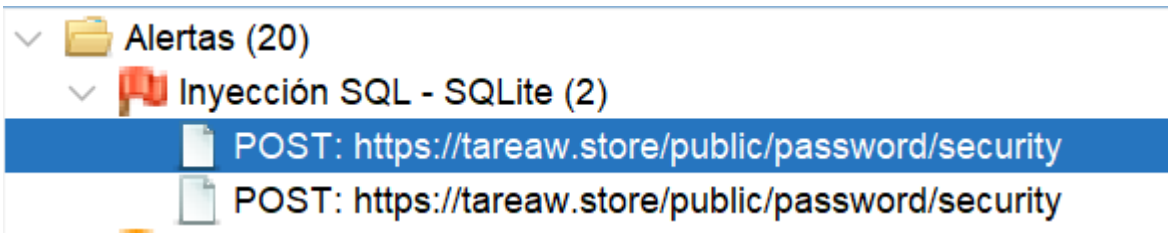
- **Vulnerabilidades y tipo que localizó la herramienta.**

  Durante el escaneo realizado con OWASP ZAP, se detectaron las siguientes vulnerabilidades.

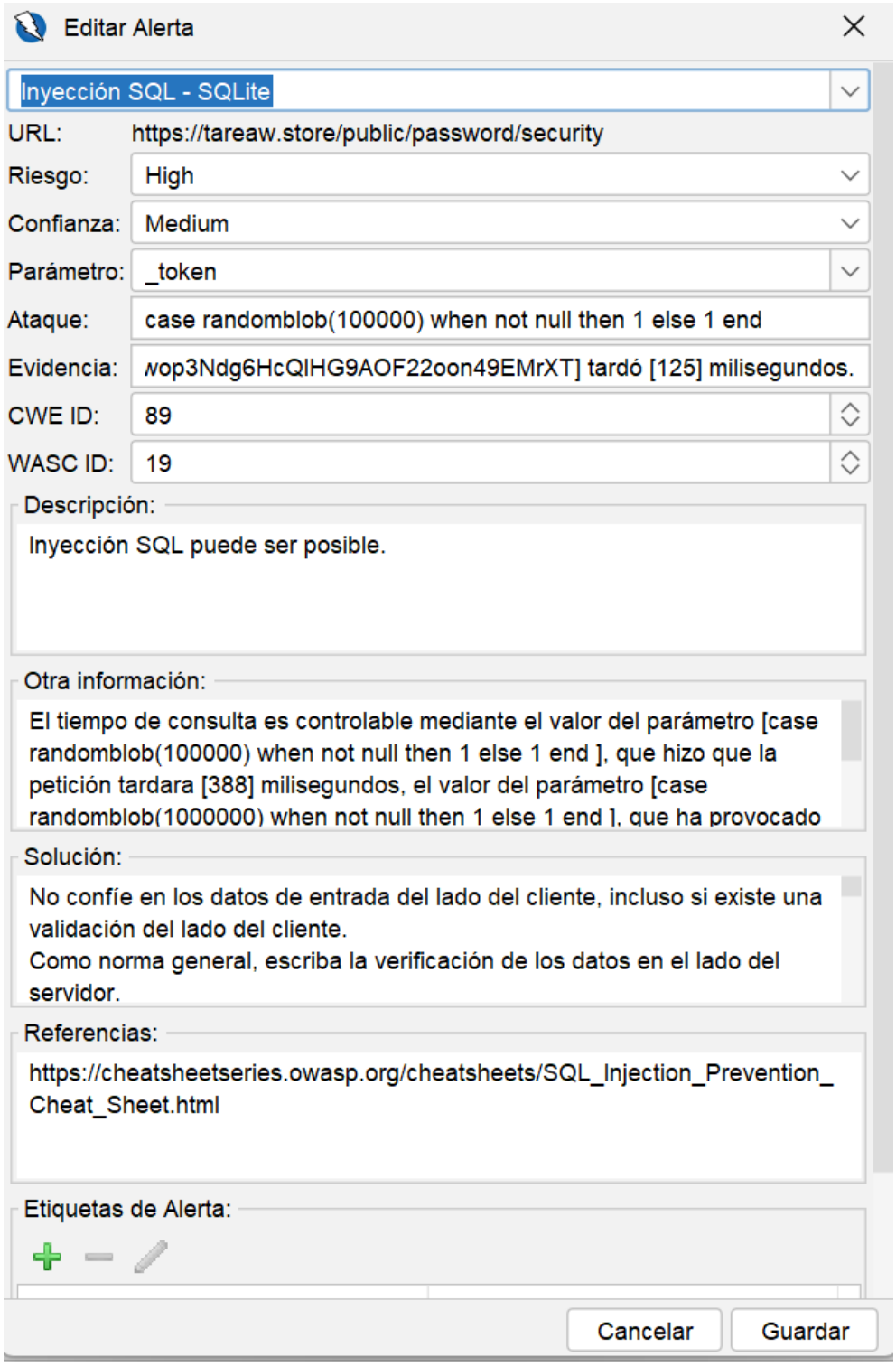
  En total fueron 20 alertas o vulnerabilidades, lo que representa riesgos potenciales para los usuarios de la aplicación y pueden ser explotadas por atacantes si no se corrigen a tiempo.



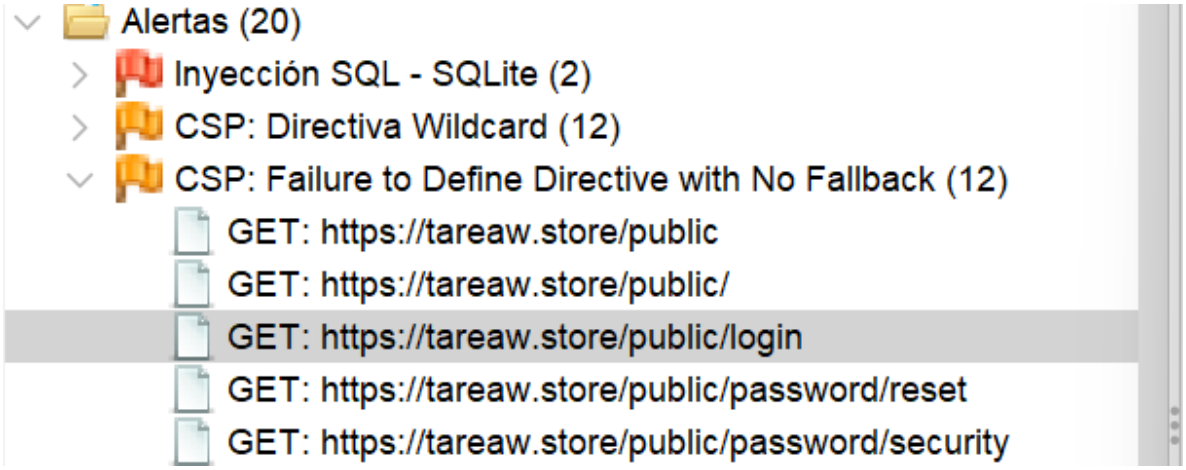
  En esta captura se observa una alerta generada por OWASP ZAP correspondiente a una vulnerabilidad de tipo SQL Injection, específicamente en una petición con método POST. Este tipo de vulnerabilidad permite a un atacante inyectar sentencias SQL maliciosas en campos de entrada de la aplicación, las cuales son ejecutadas por el servidor sin validación adecuada.



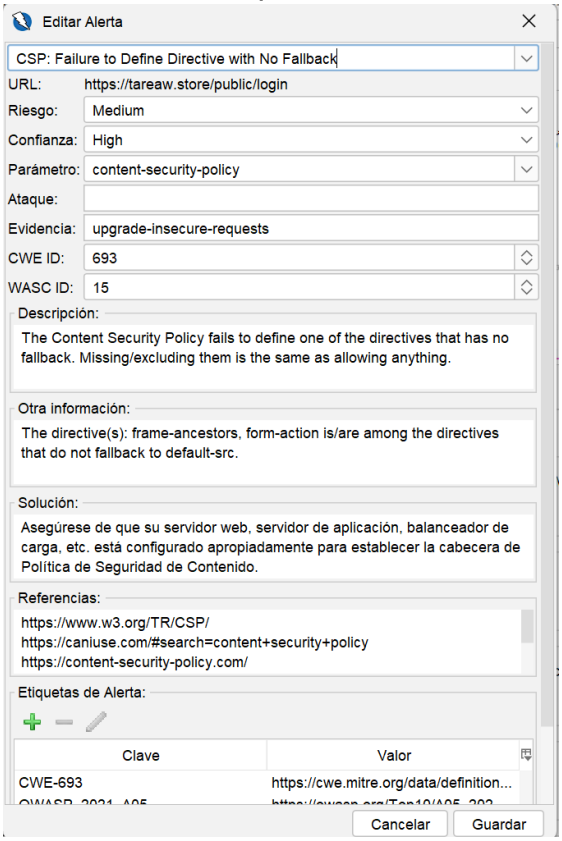
Al hacer doble clic en la alerta, se abrió una ventana con la información detallada de la petición y la respuesta del servidor. Esta ventana es clave para entender el contexto de la vulnerabilidad, lo más importante es que brinda una solución.



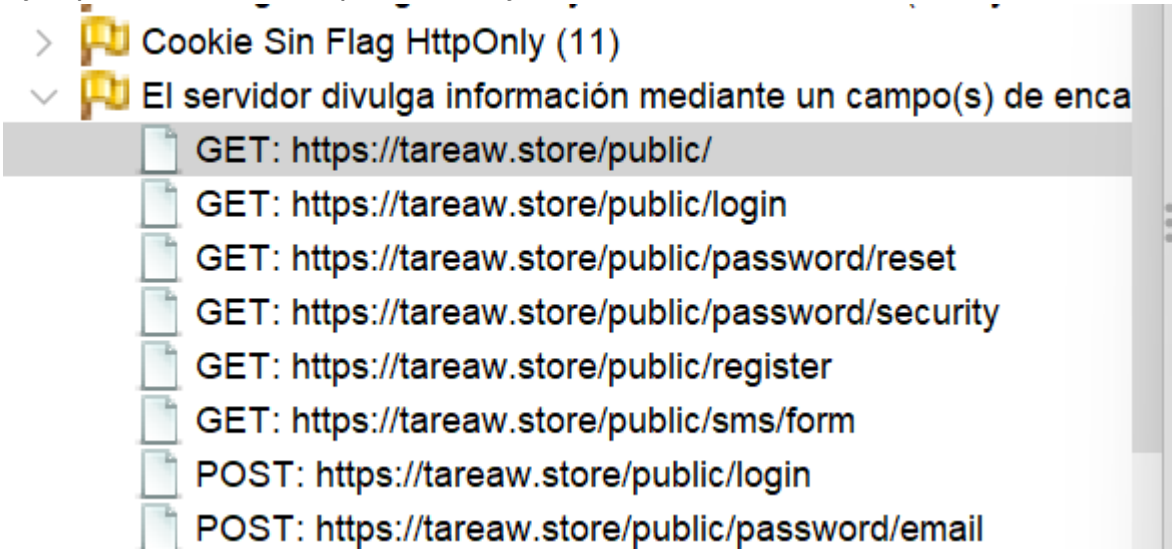
Ejemplos con alertas de prioridad media, básicamente se repite el proceso anterior para poder visualizar la información o la posible solución que nos brinda la herramienta.



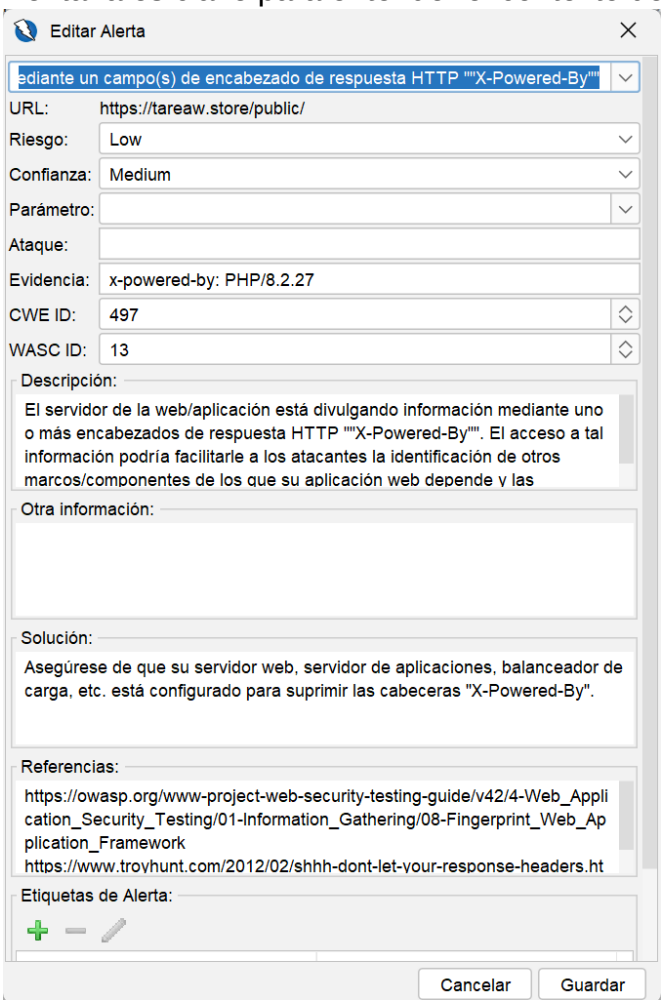
Ventana es clave para entender el contexto de la vulnerabilidad.



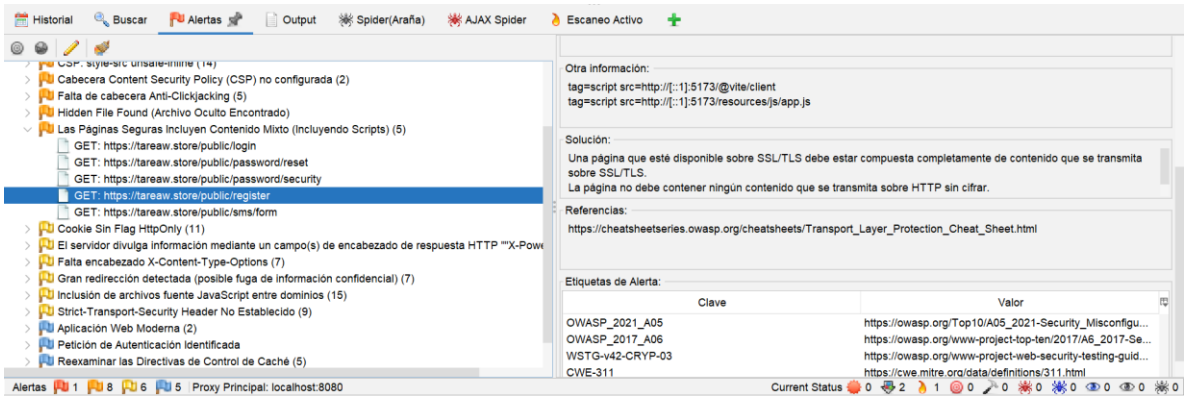
Ejemplo con una alerta prioridad baja



Ventana es clave para entender el contexto de la vulnerabilidad.



- **Interpretación de la prueba realizada con la herramienta.**



Se detectó una vulnerabilidad de tipo Reflected XSS en el endpoint.

GET: https://tareaw.store/public/register

Esto significa que el servidor refleja parte del input del usuario en la respuesta HTML sin validarlo o sanearlo adecuadamente, lo que permite que un atacante inyecte código malicioso (como JavaScript).

Un atacante podría engañar a un usuario para que haga clic en un enlace malicioso, y sin saberlo, este ejecutará código JavaScript en su navegador. Esto podría:

- Robar cookies de sesión.
- Redirigir a sitios fraudulentos.
- Modificar la apariencia de la página.
- Realizar acciones en nombre del usuario si está autenticado.

Conclusión

Esta práctica permitió explorar y aplicar una herramienta profesional de análisis de seguridad web como lo es OWASP ZAP. A través del escaneo activo y pasivo, se identificaron vulnerabilidades reales en una aplicación web de prueba. Gracias a esta experiencia, se comprendieron los riesgos de seguridad más comunes como XSS, CSRF, y la mala configuración de cookies y encabezados HTTP.

También se reforzó el conocimiento sobre la importancia de proteger las aplicaciones web desde el desarrollo, aplicando buenas prácticas y realizando pruebas constantes de seguridad.