UNIVERSIDAD NACIONAL DE SAN JUAN

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Departamento de Informática

Trabajo Final

Tecnicatura Universitaria en Programación Web

Título: "Aplicación de herramientas para el análisis estático y dinámico de código en aplicaciones web"

Alumno: Emmanuel Reynaldo Páez Poblete

DNI: 36.032.516

Asesora: Mag. Ilda Flavia Millán Tejada

Índice

Capítulo I

1.1. Introducción

El presente trabajo, para la obtención del título Técnico Universitario en Programación Web, aborda un proceso de revisión de herramientas destinado a comprobar código fuente, permitiendo identificar los errores, en el desarrollo de software. Un proceso de revisión de código, se lleva a cabo típicamente antes de fusionarse con el código base. Una revisión efectiva del mismo, evita que los errores y fallos se introduzcan en la aplicación, mejorando la calidad del código en una fase temprana del proceso de desarrollo del software. Las revisiones de los códigos conducen a mejorar la experiencia de todos los integrantes del equipo de desarrollo. Para el presente trabajo, se introducirá en el proceso de revisión de código, asistido por herramientas de software. Esto implica el uso de software especializado para facilitar el proceso de revisión, tanto en análisis estático, como dinámico. El conocer y dominar este tipo de herramientas, proporciona ayuda en tareas como:

- Organizar y mostrar los cambios actualizados en los archivos de trabajo.
- Facilitar el diálogo entre los distintos actores del proceso de desarrollo de software y los revisores de código.
- Evaluar la eficacia del proceso de revisión del código con métricas obtenidas de las herramientas de software aplicadas.

Es preciso aclarar que el software inseguro está debilitando las finanzas, salud, defensa, energía, y otras infraestructuras críticas. A medida que el software se convierte en algo crítico, complejo e interconectado, la dificultad de lograr seguridad en las aplicaciones aumenta exponencialmente. Cabe mencionar, que el principal resultado de un proceso de revisión del

código, es aumentar la eficiencia, para ello, se pretende investigar y probar herramientas de software que posibiliten realizar análisis eficientes, para lograr códigos sin errores.

En la actualidad, el análisis automatizado se ha convertido en una parte integral del proceso de desarrollo de software. Con el aumento de la complejidad del software y la necesidad de iteraciones rápidas, se busca soluciones que permitan mejorar la calidad y la seguridad del código de manera eficiente, incluyendo tendencias, herramientas y mejores prácticas.

1.2. Objetivos

1.2.1. Objetivo general:

Descubrir las vulnerabilidades de seguridad en desarrollos web, aplicando herramientas de software para análisis de código dinámico y estático y así obtener aplicaciones web seguras.

1.2.2. Objetivos específicos:

- Estudiar la problemática de la seguridad en aplicaciones web según Proyecto Abierto de Seguridad de las Aplicaciones Web (OSWAP), para conocer de una organización mundial los problemas de seguridad detectados.
- Investigar sobre técnicas de análisis de código: Pruebas estáticas de seguridad de las aplicaciones y Pruebas dinámicas de seguridad de las aplicaciones para aplicarlas convenientemente según sea el caso.
- Aplicar herramientas de software para llevar a cabo análisis estático y dinámico en aplicaciones web y con los resultados de los reportes obtenidos, proponer tratamientos a las vulnerabilidades encontradas.

1.3. Justificación

Este trabajo pretende introducir y orientar el análisis de código y presentar herramientas que asistan a este proceso. Sin desentenderse de analizar y estudiar los riesgos que a nivel mundial reportan o tienen las aplicaciones web. El conocer y aplicar herramientas que ayuden al proceso de revisión de código, cabe destacar la importancia de adoptar buenas prácticas de codificación segura desde las primeras etapas del desarrollo de software para prevenir vulnerabilidades y reducir el riesgo de ataques cibernéticos [1,2].

Algunas de estas prácticas son:

- **Gestión segura de la memoria**: Utilizar funciones seguras para la gestión de memoria, como memcpy_s y strncpy_s en lugar de memcpy y strncpy, para prevenir desbordamientos de búfer y otros errores relacionados con la memoria.
- **Validación de entrada**: Validar siempre las entradas del usuario y las entradas externas para prevenir ataques de inyección, como los ataques de SQL injection y buffer overflow.
- **Gestión de errores**: Implementar una gestión adecuada de errores para evitar fugas de información sensible y evitar que los errores del programa revelen detalles importantes sobre la implementación.
- **Uso seguro de punteros y referencias**: Utilizar punteros y referencias de manera segura para prevenir la corrupción de memoria y otros errores comunes en C y C++.
- **Control de acceso adecuado**: Implementar un control de acceso apropiado para proteger los datos y recursos sensibles de acceso no autorizado.
- **Autenticación y autorización seguras**: Utilizar métodos de autenticación seguros y sistemas de autorización para garantizar que los usuarios solo puedan acceder a los recursos para los que tienen permiso.

- **Seguridad en la comunicación**: Utilizar protocolos de comunicación seguros, como SSL/TLS, para proteger la comunicación entre sistemas y cifrar los datos sensibles durante la transmisión.
- **Pruebas de seguridad**: Realizar pruebas de seguridad regulares, como pruebas de penetración y análisis estático y dinámico del código, para identificar y corregir vulnerabilidades de seguridad antes de la implementación.

Entender que la seguridad del software es una necesidad de quien o quienes desarrollan software, es vital para desarrollar y ofrecer software de calidad y esto sin dudas, puede marcar una diferencia entre el éxito y el fracaso de una aplicación, y en el tiempo, el éxito o fracaso de una empresa dedicada al desarrollo de software.

1.4. Metodología

Para llevar a cabo los objetivos y alcance del presente trabajo final, la metodología que se adopta es de tipo exploratoria, la cual permite identificar las áreas específicas de seguridad de la información para el análisis de código. Esto orienta y enfoca un estudio de herramientas dedicadas al análisis de código, y de esta forma se logra obtener los datos de prueba que servirán para su posterior análisis e interpretación. Esta investigación sobre las herramientas disponibles, libres y probadas por una comunidad de desarrollo web mundial, sirve para abordar los aspectos de seguridad informática seleccionados. Esto se logra, evaluando las características, capacidades, fortalezas y debilidades de cada herramienta, determinando su aplicación, de acuerdo a lo que se desee revisar. Luego, se propone un detalle para la implementación de las herramientas seleccionadas, incluyendo la configuración, instalación y puesta en marcha. A su vez, se realizan acciones de verificación del funcionamiento y la efectividad de las mismas, evaluando la

capacidad para detectar, prevenir o mitigar las amenazas de seguridad. Por último, se analizan los resultados de las pruebas y evaluaciones realizadas, para identificar áreas de mejora y posibles vulnerabilidades, comparando los resultados con los objetivos establecidos y de esta forma, determinar si se han cumplido satisfactoriamente, documentando todas las actividades realizadas, incluyendo la selección, implementación de las herramientas seleccionadas y los resultados de las pruebas que fueron obtenidos de las mismas.

Capítulo 2

2.1. Marco Teórico

La seguridad de la información no se preocupa sólo por el medio informático, se preocupa por todo aquello que pueda contener información, en resumen, esto quiere decir que se preocupa por casi todo, lo que conlleva a afirmar que existen varias diferencias, pero lo más relevante es el universo que manejan cada uno de los conceptos en el medio informático. Según Aguilera, se puede definir a la seguridad de la información como la disciplina encargada de plantear y diseñar las normas, procedimientos, métodos y técnicas con el fin de obtener que un sistema de información sea seguro, confiable y sobre todo que tenga disponibilidad. Por ello, es muy importante contar con aplicaciones de software seguras y libres de errores. Se considera una amenaza a cualquier evento accidental o intencional o que pueda causar algún daño en el sistema informático, provocando pérdidas materiales, financieras o de otro tipo a la organización [3]. Entre las características más relevantes de una amenaza se encuentran las siguientes [4]:

- El origen puede ser interno o externo.
- La motivación como son las ventajas competitivas, los beneficios económicos etc.
- La frecuencia o periodicidad de los ataques.
- La severidad dependiendo si es o no y reversible.

Se puede establecer la siguiente clasificación a la hora de estudiar las amenazas de seguridad [1]:

- Amenazas naturales: inundación, incendio, tormenta, falla eléctrica, explosión, etc.
- Amenazas de agentes externos: virus informático, ataques a una organización criminal, sabotaje terrorista, disturbios y conflictos sociales, intrusos en la red como robot, estafa, etcétera

- Amenazas de agentes internos: empleados descuidados, empleados con una formación inadecuada o descontentos, errores en la utilización de las herramientas y recursos del sistema.
- Una vulnerabilidad es cualquier debilidad en el sistema informático que pueda permitir a las amenazas causarle daños y producir pérdidas en la organización [5]. Las vulnerabilidades se corresponden con fallos en los sistemas físicos biológicos, aunque también pueden tener su origen en los defectos de ubicación, instalación, configuración y mantenimiento de los equipos. Las vulnerabilidades, como se mencionó anteriormente, son un defecto de una aplicación que puede ser aprovechado por un atacante. Si lo descubre, el atacante programará un software (llamado malware) que utiliza esa vulnerabilidad para tomar el control de la máquina (exploit) o realizar cualquier operación no autorizada.

Hay tres tipos de vulnerabilidades [6]:

- Vulnerabilidades reconocidas por el suministrador de la aplicación y para las cuales ya tiene un parche que las corrige. Se debe aplicar el parche inmediatamente.
- Vulnerabilidades reconocidas por el suministrador, pero todavía no hay un parche. En algunos casos sí se proporciona una solución temporal (workaround), pero, generalmente, lo mejor es desactivar el servicio hasta haber aplicado el parche.
- Vulnerabilidades no reconocidas por el suministrador. Es el peor caso, porque se puede estar expuestos a un ataque durante un tiempo largo sin saberlo.

Un incidente de seguridad es cualquier evento que tenga o pueda tener como resultado la interrupción de los servicios suministrados por un sistema informático y/o posibles pérdidas físicas, de servicios de activos o financieras. Es decir, se considera que un incidente es la

materialización de una amenaza. El impacto es una medición y valoración del daño que podría producir a la organización un incidente de seguridad. Para valorar el impacto es necesario tener en cuenta tanto los daños tangibles, como la estimación de los daños intangibles, incluida la información. En este sentido, podría resultar de gran ayuda la realización de entrevistas en profundidad con los responsables de cada departamento, función o proceso de negocio, tratando de determinar cuál es el impacto real de la revelación, alteración o pérdida de la información para la organización, y no sólo del elemento tecnológico que la soporta [7].

Es preciso tener en cuenta, que en la actualidad el software inseguro está debilitando las finanzas, salud, defensa, energía, y otras infraestructuras críticas. A medida que el software se convierte en algo crítico, complejo e interconectado, la dificultad de lograr seguridad en las aplicaciones aumenta exponencialmente. El ritmo vertiginoso de los procesos de desarrollo de software actuales, incrementa aún más el riesgo de no descubrir vulnerabilidades de forma rápida y precisa [8]. Por ello, es preciso estudiar los incidentes de seguridad a nivel mundial, para determinar las vulnerabilidades del software desarrollado. Luego aplicar herramientas de software para asistir el proceso de revisión de código. Para esto, existen varios métodos de análisis de riesgos que sirven para proteger la información valiosa de una organización. Los que se abordarán en este trabajo son dos: Pruebas estáticas de seguridad de las aplicaciones y Pruebas dinámicas de seguridad de las aplicaciones.

2.1.1. Seguridad de la información

La seguridad de la información se refiere a la protección de la información y sus sistemas de hardware, software y comunicaciones contra el acceso, divulgación, alteración, destrucción o interrupción no autorizados, tanto en almacenamiento como en tránsito [9].

La importancia de proteger la información en todas sus formas y en todas las etapas de su ciclo de vida, así como los sistemas que la almacenan, procesan y transmiten. La seguridad de la información abarca una amplia gama de medidas y controles diseñados para garantizar la confidencialidad, integridad y disponibilidad de la información, así como para protegerla contra una variedad de amenazas y riesgos en el entorno digital actual.

2.1.2. Aplicaciones Web

Aunque los inicios de internet se remontan a los años setenta, no ha sido hasta los años noventa cuando, gracias a la web, se ha extendido su uso por todo el mundo. En pocos años la Web ha evolucionado enormemente; se ha pasado de páginas sencillas, con pocas imágenes y contenidos estáticos a páginas complejas con contenidos dinámicos que provienen de bases de datos, lo que permite la creación de aplicaciones web.

Una aplicación web se puede definir como una aplicación con la cual el usuario por medio de un navegador realiza peticiones a una aplicación remota accesible a través de Internet y que recibe una respuesta que se muestra en el propio navegador.

Una aplicación web (web-based application) es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (HTTP - Hypertext Transfer Protocol)[1] están estandarizados y no han de ser creados por el programador de aplicaciones. (Figura 1) [10].

[1] HTTP: (Protocolo de transferencia de hipertexto) El Protocolo de transferencia de hipertexto (HTTP) es la base de la World Wide Web y se utiliza para cargar páginas web mediante enlaces de hipertexto. Es un protocolo cliente-servidor en el que un cliente (como un navegador web) envía una solicitud a un servidor (como un sitio web), y el servidor responde con un recurso (como una página web).

El protocolo HTTP forma parte de la familia de protocolos de comunicaciones TCP/IP, que son los empleados en Internet. Estos protocolos permiten la conexión de sistemas heterogéneos, lo que facilita el intercambio de información entre distintos ordenadores.

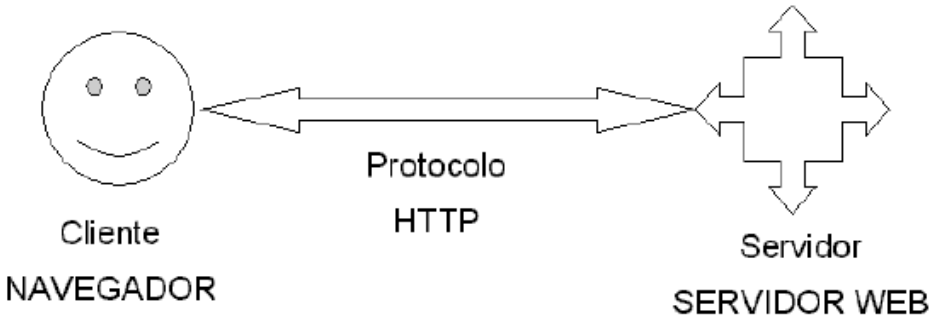


Figura 1: Esquema básico de una aplicación web. Autor: Sergio Lujan.

- El **cliente** web es un programa con el cual interacciona el usuario para solicitar a un servidor web el envío de los recursos que desea obtener mediante HTTP. La parte cliente de las aplicaciones web suele estar formada por el código HTML que forma la página web más algo de código ejecutable realizado en lenguaje de script del navegador (como JavaScript) o mediante pequeños programas (applets) realizados en Java.

  Las tecnologías que se suelen emplear para programar el cliente web son:

  - HTML.
  - CSS.
  - DHTML.
  - Lenguajes de script: JavaScript, VBScript, etc.
  - ActiveX.
  - Applets programados en Java.

- Distintas tecnologías que necesitan la existencia de un plug-in en el navegador: Adobe Acrobat Reader, Autodesk MapGuide, Live Picture PhotoVista, Macromedia Flash, Macromedia Shockwave, Virtual Reality Modeling Language (VRML), etc.

- Un **servidor** web es un programa o software diseñado para procesar y responder a las solicitudes de los clientes web. Su función principal es almacenar, procesar y servir contenido web a través de internet. Cuando el cliente web, como un navegador, realiza una solicitud para acceder a una página web, un archivo u otro recurso en línea, el servidor web recibe esa solicitud, la procesa y envía la respuesta correspondiente de vuelta al cliente.
  - Los servidores web utilizan protocolos de comunicación estándar, como HTTP (Hypertext Transfer Protocol) o HTTPS (HTTP Secure), para transferir datos entre el servidor y el cliente. Estos protocolos establecen reglas y procedimientos para la transferencia segura de datos a través de internet.
  - Pueden alojar una amplia variedad de contenido, incluyendo páginas web estáticas, aplicaciones web dinámicas, archivos multimedia, servicios en línea y mucho más. Además, pueden ejecutar diferentes tipos de software y tecnologías, como PHP, Python, Node.js, Java, entre otros, para procesar y generar contenido dinámico según las solicitudes de los clientes.
  - Las bases de datos en el servidor web es una herramienta fundamental para almacenar y gestionar datos de manera eficiente en aplicaciones web. Puede contener información de usuarios, contenido de la aplicación, datos

transaccionales, configuraciones y más. Algunos tipos de bases de datos pueden ser: MySQL, PostgreSQL, SQL Server.

La seguridad en el lado del cliente y del servidor en las aplicaciones web es un tema crítico en el desarrollo de software. Por lo que comunidades como OWASP, u Open Web Application Security Project, se dedican a mejorar la seguridad del software. "OWASP Testing Guide", esta guía proporciona una colección de técnicas y pruebas para evaluar la seguridad en aplicaciones web desde el lado del cliente y del servidor.

2.1.3. OWASP

OWASP (acrónimo de Open Web Application Security Project - Proyecto abierto de seguridad en aplicaciones web) es una iniciativa de código abierto dedicada a determinar y combatir las causas que hacen que el software sea inseguro.

Alrededor de la Fundación OWASP hay empresas, organizaciones educativas y también miles de personas de todo el mundo. Estos agentes constituyen la que se conoce como comunidad OWASP. Su fin es crear material didáctico (artículos de divulgación, metodologías, ejemplos de buenas prácticas, documentación, manuales, etc.) y también herramientas y tecnologías. Con todos estos recursos, que pueden usarse de forma libre y sin coste, pretenden promover un entorno tecnológico más seguro [11]

Los objetivos que persigue esta iniciativa son:

- Promover la seguridad de las aplicaciones web.
- Buscar las causas de la inseguridad en el software.
- Proporcionar soluciones a las amenazas de las aplicaciones.
- Ayudar a las empresas a entender la seguridad de las aplicaciones web y desarrollar aplicaciones más seguras.
- Crear herramientas, documentación y estándares open-source.

Al seguir las recomendaciones de OWASP, se hace uso de sus herramientas y tecnologías, y, en definitiva, si se adopta una metodología de desarrollo seguro, se consigue la reducción general del riesgo en una organización y el aumento de su confianza. Asimismo, se destacan los siguientes beneficios:

- Se obtiene un software más seguro, ya que, al haberse considerado la seguridad desde un primer momento, permite detectar y solucionar tempranamente los fallos de seguridad.
- Se reducen los costos, ya que las modificaciones, el tiempo y la complejidad de la solución de los fallos de seguridad que se detectan son menores.
- Se identifican causas y evitan que se repitan errores comunes de seguridad, por lo que se ve reducida la probabilidad de vulnerabilidades en el producto final.
- Se mitiga el impacto de la explotación de vulnerabilidades no detectadas en el ciclo de desarrollo.
- Se genera conciencia de seguridad en los equipos involucrados en el ciclo de desarrollo.
- Se dispone de una medición tangible sobre el estado de la seguridad en el software, lo que permite la toma de decisiones informadas relacionadas con la seguridad, así como su conocimiento y registro.

De entre los proyectos desarrollados por la comunidad de OWASP, se destacan los siguientes [12]:

- **OWASP Top 10**: Es un informe que recoge las vulnerabilidades más comunes dentro de las aplicaciones web y cómo prevenirlas. El más reciente enumera y detalla las siguientes: errores de inyección, autenticación y gestión de sesiones, XSS, control de acceso, error de configuración de seguridad, exposición de datos sensibles, protección insuficiente ante los ataques, CSRF, utilizar componentes con vulnerabilidades conocidas y APIs desprotegidas. En definitiva, crea conciencia sobre los desafíos que enfrentan las organizaciones, lo que garantiza la seguridad de aplicaciones web en un entorno que cambia rápidamente.
- **OWASP Proactive Controls (Controles proactivos)**: Es un documento que resume las buenas prácticas que deberían incluirse en cada proyecto de desarrollo de software. La lista de controles, por orden de importancia, es:

1. Verificar la seguridad desde el inicio y a menudo.
2. Parametrizar las consultas.
3. Codificar los datos.
4. Validar todas las entradas.
5. Implementar controles de autenticación y de identidad.
6. Implementar controles de acceso apropiados.
7. Proteger los datos.
8. Implementar el registro y la detección de intrusos.

9. Aprovechar los frameworks y las librerías de seguridad.
10. Manejo de errores y excepciones.

- **OWASP Application Security Verification Standard:** (ASVS - Estándar para la Verificación de la Seguridad en Aplicaciones es español**):** Es un estándar para la realización de comprobaciones de seguridad a nivel de aplicación.

2.1.3.1. Niveles de riesgo de OWASP

OWASP define tres niveles de riesgo:

1. Nivel 1: pensado para todo tipo de software.
2. Nivel 2: pensado para aplicaciones que contienen información sensible que requiere protección.
3. Nivel 3: pensado para las aplicaciones más críticas – aplicaciones que realizan transacciones de gran valor, contienen información médica sensible o cualquier aplicación que requiere el mayor grado de confianza.

Cada nivel contiene una lista de requisitos que deben ser mapeados en software para cubrir los requerimientos de seguridad necesarios.

Los requerimientos de ASVS fueron desarrollados con los siguientes objetivos en mente:

1. Utilizar como una métrica: provee a los desarrolladores y gerentes de aplicaciones con una métrica para determinar el nivel de confianza de las mismas.
2. Utilizar como una guía: provee a los desarrolladores de controles de seguridad con indicaciones sobre qué funcionalidades incluir para cumplimentar con los requerimientos de seguridad

3. Utilizar durante adquisiciones: provee una base para especificar los requerimientos de seguridad en aplicaciones adquiridas a terceros.

- **OWASP "Cheat Sheet":** conjunto de guías de buenas prácticas, tanto de prevención como de protección, para tener a mano de forma rápida la información de seguridad necesaria para aplicar las técnicas correctas cuando se está ensamblando y programando las rutinas esenciales de las aplicaciones. Ejemplos: cheat sheet para prevenir ataques de XSS, cheat sheet para prevención de ataques de inyección SQL, cheat sheet sobre almacenamiento criptográfico, etc.

2.1.3.2. Herramientas

OWASP también realiza una serie de proyectos con el objetivo de proporcionar herramientas para afianzar la seguridad de las aplicaciones. A continuación, se muestran algunos de estos proyectos junto con una breve descripción de cada uno de ellos:

- Zed Attack Proxy: herramienta de pentesting web que ayuda a encontrar vulnerabilidades en las aplicaciones.
- Dependency Check: herramienta que permite analizar todas las dependencias de las aplicaciones y comprobar si existen vulnerabilidades dentro de ellas.
- Mobile Security Project: herramienta que permite realizar pentesting sobre aplicaciones móviles.
- SSL advanced forensic tool: herramienta que permite mostrar información sobre SSL y los certificados.

- WebGoat: aplicación educativa que provee una base de prueba para encontrar fallos de seguridad. Sirve para aprender sobre ataques de XSS, SQL Injection, robo de sesión, inyección XPath, etc.
- WebScarab: framework para analizar tráfico HTTP/HTTPS, observando el tráfico entre el navegador y el servidor, exponiendo campos ocultos, análisis y recopilación de cookies, etc.

2.1.4. Análisis de código

La calidad y la seguridad del software se pueden mejorar mediante un proceso llamado análisis del código fuente, que implica un examen exhaustivo del código fuente del programa en busca de errores, fallas de seguridad y otros problemas. El análisis se puede realizar manualmente o con la ayuda de herramientas automáticas que escanean el código en busca de posibles fallas y producen hallazgos para una mayor investigación. A menudo se realiza como parte del ciclo de vida de desarrollo de software [13].

El propósito fundamental del análisis del código fuente es descubrir posibles errores ayudando a identificar vulnerabilidades de seguridad en el código antes de que creen problemas en el mundo real. Es posible mejorar la calidad del software y reducir el riesgo de vulnerabilidades de seguridad detectando y corrigiendo fallas en las primeras etapas del proceso de desarrollo.

El análisis del código fuente también puede ayudar a mantener el código y reducir costos al encontrar lugares donde el código pueda modificarse o refactorizar para aumentar el rendimiento, la mantenibilidad o la legibilidad [14].

Existen varios métodos de análisis de riesgos y vulnerabilidades que sirven para proteger la información por lo que se encuentran herramientas de seguridad DAST, SAST y SCA cumplen diferentes parámetros de protección durante el desarrollo [15]:

- **SAST: Análisis Estático (Static Application Security Testing)** es una herramienta de análisis estático que evalúa el código fuente de una aplicación para identificar vulnerabilidades de seguridad antes de que se implemente en un entorno de producción real.
- **DAST: Análisis Dinámico (Dynamic Application Security Testing)** es una prueba automatizada que evalúa la seguridad de una aplicación web en tiempo de ejecución.
- **SCA: Análisis de Composición (Software Composition Analysis)** son herramientas de seguridad que se enfocan en escanear el código fuente y/o el paquete de distribución de la aplicación para identificar las bibliotecas y componentes de terceros que se utilizan. Luego, las herramientas SCA analizan estas bibliotecas y componentes para identificar vulnerabilidades conocidas de seguridad, así como para evaluar la calidad del software y cumplimiento de políticas de seguridad.

Comunidades como OWASP, el Proyecto Abierto de Seguridad de Aplicaciones Web, ofrece una guía exhaustiva sobre cómo llevar a cabo el análisis de código para identificar y mitigar las vulnerabilidades comunes en las aplicaciones web. Su visión general de abordar el análisis de código estático y dinámico.

2.1.4.1. Análisis estático (Static Application Security Testing – SAST)

El análisis de código estático (también conocido como análisis de código fuente) generalmente se realiza como parte de una revisión de código y se lleva a cabo en la fase de implementación de

un ciclo de vida de desarrollo de seguridad (SDL). El análisis de código estático comúnmente se refiere a la ejecución de herramientas de análisis de código estático que intentan resaltar posibles vulnerabilidades dentro del código fuente "estático" (no en ejecución) mediante el uso de técnicas como el análisis de contaminación y el análisis de flujo de datos.

Dichas herramientas encontrarán automáticamente fallas de seguridad con un alto grado de confianza de que lo que se encuentra es efectivamente una falla. Por lo tanto, frecuentemente sirven como ayuda para que un analista pueda concentrarse en partes del código relevantes para la seguridad pudiendo encontrar fallas de manera más eficiente, en lugar de una herramienta que simplemente encuentra fallas automáticamente.

Cabe destacar yendo más allá de cómo funcionan estas herramientas, están compuestas por mecanismos y técnicas, pudiendo por medio de su implementación lograr extraer información del código analizado, por lo que será sin duda información relativa a la seguridad del código auditado. Estás técnicas pueden ser [16]:

- **Grep análisis**: (Comando Grep) Utilidad para el uso de regular expresión —palabras reservadas— detección de contraseñas visibles.
- **Identificador de código**: Utilidad para verificar que una vez que se abren llaves esté bien identado el código —esto puede evitar fallos de seguridad, como por ejemplo apple-go-tofail.
- **DataFlowAnalysis**: Análisis de flujo de datos, es una utilidad para, desde un punto de partida, encontrar la pertenencia con un punto final.
- **Propagación de constante**: Utilidad para, teniendo una constante definida en un sitio, ver cómo se propagaría esa constante dentro del código fuente.
- **Alias análisis**: Utilidad para hacer análisis de punteros.

- **Análisis de Dependencias (SCA):** Aunque puede considerarse parte del análisis estático, se centra en las dependencias externas utilizadas por el proyecto. Este tipo de análisis examina bibliotecas y paquetes para detectar vulnerabilidades conocidas y problemas de seguridad.

Para esta tarea OWASP TOP 10 se vuelve un marco de referencia excelente, ya que contempla la evaluación de las vulnerabilidades más relevantes del software. Si bien es cierto que se dispone de las herramientas de tipo SAST para el análisis estático de código, al final este tipo de herramientas están más orientadas a la calidad de código que a la seguridad.

2.1.4.2. Análisis Dinámico (Dynamic Application Security Testing - DAST)

El análisis de código dinámico, también llamado Prueba dinámica de seguridad de aplicaciones (DAST), está diseñado para probar una aplicación en ejecución en busca de vulnerabilidades potencialmente explotables. Herramientas DAST para identificar vulnerabilidades tanto en tiempo de compilación como en tiempo de ejecución, como errores de configuración que solo aparecen dentro de un entorno de ejecución. Estas herramientas DAST utilizan un diccionario de vulnerabilidades conocidas y entradas maliciosas para escanear una aplicación.

Algunas de estas entradas potencialmente maliciosas pueden ser [17]:

- Consultas SQL (para identificar vulnerabilidades de inyección SQL)
- Cadenas de entrada largas (para explotar la vulnerabilidad de desbordamiento de buffer)
- Números negativos y positivos grandes (para detectar vulnerabilidades de desbordamiento y desbordamiento de enteros)
- Datos de entrada inesperados (para explotar suposiciones no válidas por parte de los desarrolladores)

A medida que la aplicación se ejecuta, es bombardeada por estas entradas potencialmente maliciosas y la herramienta DAST analiza las respuestas de la aplicación. Si la aplicación tiene una respuesta negativa a una entrada (como fallar, devolver una respuesta no válida, etc.), entonces la herramienta DAST registra la vulnerabilidad identificada.

Dado que las herramientas DAST se ejecutan en una aplicación en ejecución, pueden detectar una amplia gama de vulnerabilidades potenciales. Esto incluye vulnerabilidades que son difíciles o imposibles de detectar en el código fuente, como problemas de asignación de memoria.

DAST generalmente entra en juego en la fase de prueba del SDLC. Esto se debe a que DAST requiere la capacidad de ejecutar la aplicación y probarla utilizando entradas maliciosas simuladas. Como resultado, una vez que el código de la aplicación pueda compilarse e implementarse en un entorno de prueba o de ensayo, puede utilizar DAST. Con flujos de trabajo de integración continua/entrega continua (CI/CD)[2], eso puede significar que los escaneos DAST se ejecuten varias veces a medida que se producen compilaciones iterativas.

OWASP proporciona varias herramientas que se utilizan para el análisis dinámico de aplicaciones, especialmente en el contexto de pruebas de penetración y seguridad web. Aquí están algunas de las herramientas OWASP más relevantes para el análisis dinámico:

- OWASP ZAP (Zed Attack Proxy): ZAP es una de las herramientas más conocidas de OWASP para pruebas de penetración y análisis dinámico de aplicaciones web. Permite a los usuarios interceptar y modificar tráfico HTTP, escanear aplicaciones web para detectar vulnerabilidades, y simular ataques para identificar posibles riesgos de seguridad.

[2] CI/CD: (Integración y entrega continua) Es una práctica de desarrollo de software que automatiza la construcción, prueba e implementación de código, permitiendo una entrega más rápida y eficiente de software.

- OWASP AppSensor: Este proyecto OWASP se centra en la detección y respuesta a ataques en tiempo real. Aunque no es análisis dinámico en el sentido tradicional, proporciona un marco para incorporar sensores y mecanismos de detección en aplicaciones web para detectar comportamientos anómalos o maliciosos durante la ejecución.
- OWASP WebScarab: Aunque ya no es tan popular como ZAP, WebScarab era una herramienta de análisis dinámico y pruebas de penetración para aplicaciones web. Permitía a los usuarios interceptar y analizar el tráfico HTTP y HTTPS.

Capítulo 3

2.2. Herramientas estáticas y dinámicas

OWASP es conocido por su compromiso con la mejora de la seguridad en aplicaciones web y proporciona una variedad de herramientas para análisis estático y dinámico.

Entre las más conocidas se encuentran OWASP ZAP para el análisis dinámico y OWASP Dependency Check para el análisis estático.

2.2.1. Escaneo de vulnerabilidades automático con OWASP ZAP - Análisis dinámico

OWASP ZAP (Zed Attack Proxy) es una de las herramientas de seguridad de aplicaciones web de código abierto más populares y ampliamente utilizadas. Es una herramienta de prueba de penetración que se utiliza para encontrar vulnerabilidades en aplicaciones web durante el desarrollo y la prueba de seguridad. ZAP puede ayudar a identificar problemas de seguridad comunes, como inyecciones de SQL, cross-site scripting (XSS), secuencias de comandos entre sitios (CSRF) y más.

Además de encontrar vulnerabilidades, ZAP también puede utilizarse para realizar pruebas de seguridad automatizadas, así como para interceptar y modificar el tráfico entre el navegador web y la aplicación web objetivo, lo que permite realizar pruebas de seguridad manuales más avanzadas [18].

- *Para su instalación, ver Anexo de instalación (Pág. 113)*

2.2.1.1. Sesiones en ZAP

Al iniciar ZAP se podrá observar un cuadro de diálogo donde pregunta si se quiere conservar las sesiones que se trabajará (Figura 2).

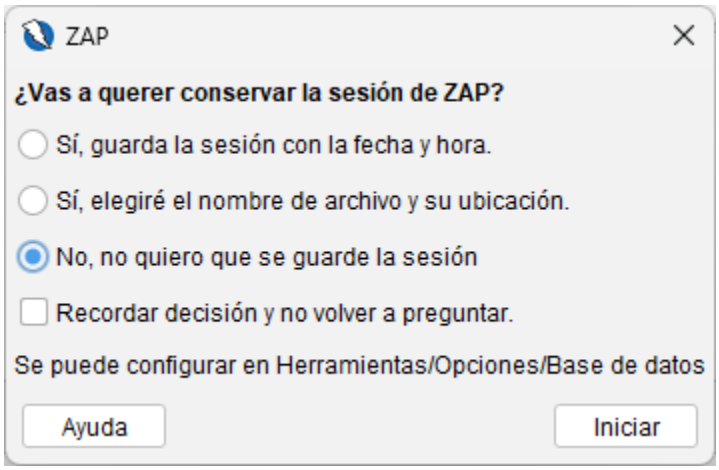


Figura 2: Cuadro de diálogo para conservar la sesión de ZAP – Diseño del autor.

Al seleccionar la primera opción, ZAP guardará la sesión con la fecha en la que se inicia, significa que se podrá guardar en una base de datos local donde se tendrá acceso en una fecha posterior. Además, contará con un autoguardado por lo que no es necesario guardarla continuamente.

La segunda opción abre una ventana donde se podrá acceder a la ubicación de una sesión guardada anteriormente y continuar con las tareas.

Al elegir la tercera opción no creará un archivo de guardado, ni se podrá acceder a uno, lo cual procederá a iniciar el programa. Posteriormente podrá ser guardada desde el menú de archivos.

2.2.1.2. Interfaz inicial

Al iniciar la herramienta se podrá ver su ventana principal, con sus áreas principales (Figura 3).

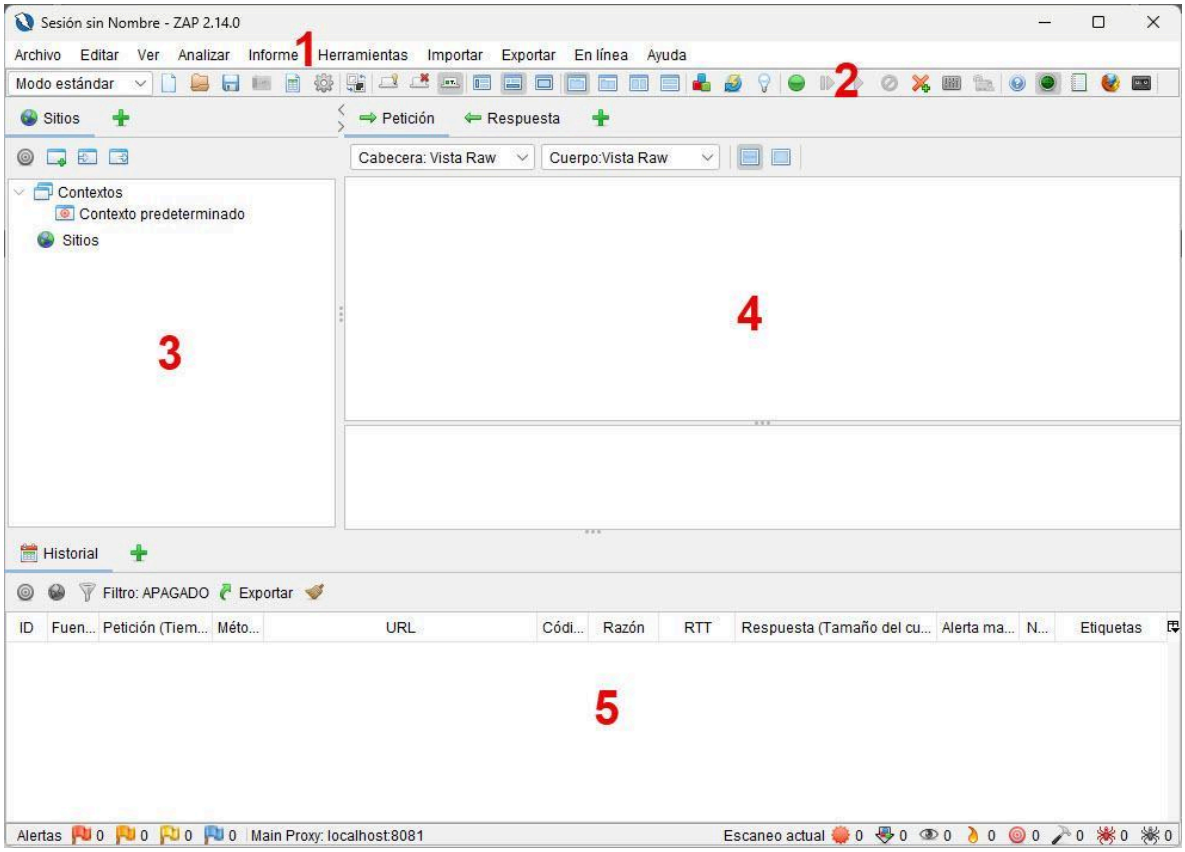


Figura 3: Ventana principal de la aplicación - Diseño del autor.

Estás áreas son:

1. **Barra de menú:** Contiene una serie de opciones desplegables que representan diferentes funciones y características disponibles en el programa.
    a. **Archivos**: Maneja las sesiones, como la actual.
    b. **Editar**: Maneja los resultados, seleccionando los modos y realiza búsquedas.
    c. **Ver**: Muestra las opciones de visualización.
    d. **Análisis**: Se encarga de las políticas de escaneo.
    e. **Informes**: Genera informes.
    f. **Herramientas**: Muestra las opciones generales del programa.
    g. **Importar**: Importación de datos a la aplicación.
    h. **Online**: Accede a recursos en línea.

i. **Ayuda**: Sector de ayuda para la aplicación.

2. **Barra de herramientas:** Se encuentran las acciones más utilizadas.
3. **Árbol:** Muestra una estructura jerárquica de los sitios web que se están escaneando en el panel "Sitios" o "Sitios y contextos". Este panel muestra una lista de los sitios web y aplicaciones que ZAP ha visitado o está visitando durante una sesión. Puedes expandir cada sitio para ver los diferentes nodos y recursos que se han encontrado, como URLs, parámetros, cookies, entre otros.
4. **Espacio de trabajo:** Es un entorno donde se puede organizar y gestionar las sesiones. Muestra las solicitudes y las respuestas que se realizaron.
5. **Ventana de Información:** Proporciona detalles y contexto sobre los elementos seleccionados en la interfaz de usuario. Esta ventana es una herramienta útil para comprender mejor los resultados de las pruebas de seguridad web realizadas por ZAP y para tomar decisiones informadas sobre cómo abordar las vulnerabilidades detectadas: como el historial y la búsqueda de solicitudes y respuestas, además de las alertas.

2.2.1.3. Creación de certificados SSL Dinámicos.

En OWASP ZAP, se generan certificados SSL[3] dinámicos para facilitar la interceptación y el análisis del tráfico HTTPS durante las pruebas de seguridad. Para su configuración se debe dirigir a la pestaña "Herramientas" y seleccionar "Opciones", en el menú lateral izquierdo, elegir la opción "Certificados SSL Dinámicos". A continuación, se debe hacer clic en "Generar" para crear un nuevo certificado raíz, si no se dispone de uno ya configurado, se debe seguir las instrucciones para generar un certificado raíz y asegúrate de exportarlo si lo necesitas para

[3] Certificados SSL: Se utilizan para establecer una conexión segura entre un navegador web y un servidor.

instalarlo en el navegador u otros dispositivos. Además, se podrá importar uno ya generado anteriormente (Figura 4).

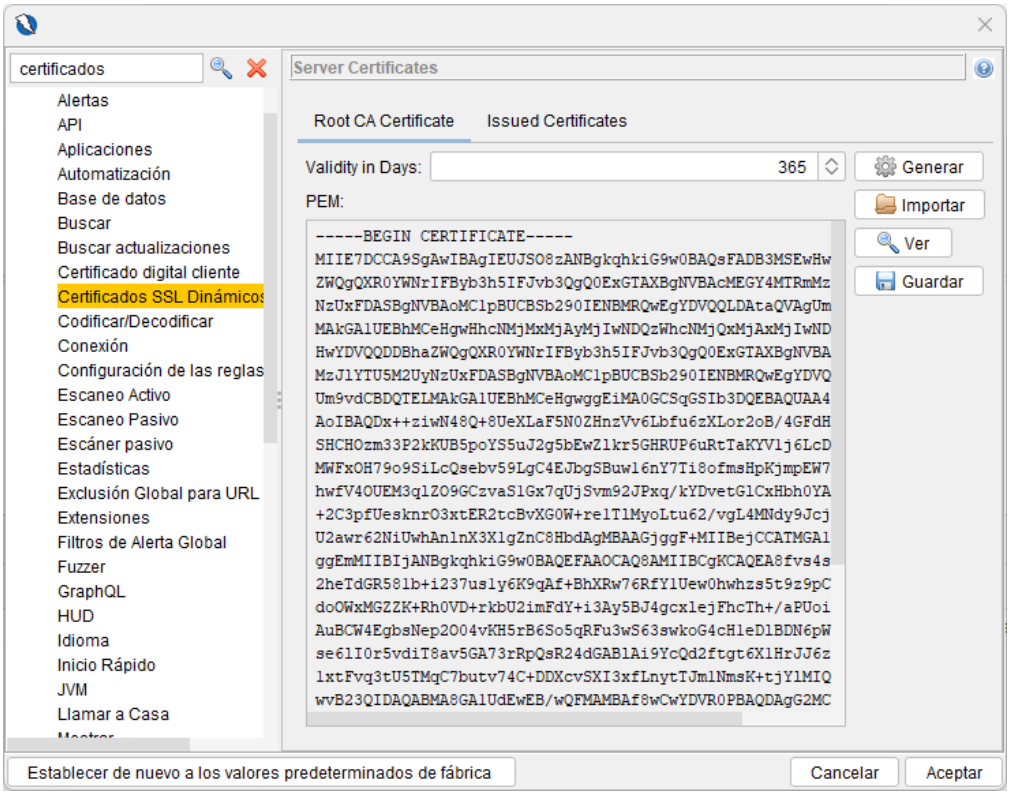


Figura 4: Certificado SSL Dinámicos en las opciones de la aplicación - Diseño del autor.

Para habilitar el Proxy SSL en la aplicación, se debe dirigir a la pestaña "Herramientas", "Opciones". En el menú lateral izquierdo, ingresar a la sección "Network", “Local Servers/Proxies” (Figura 5).

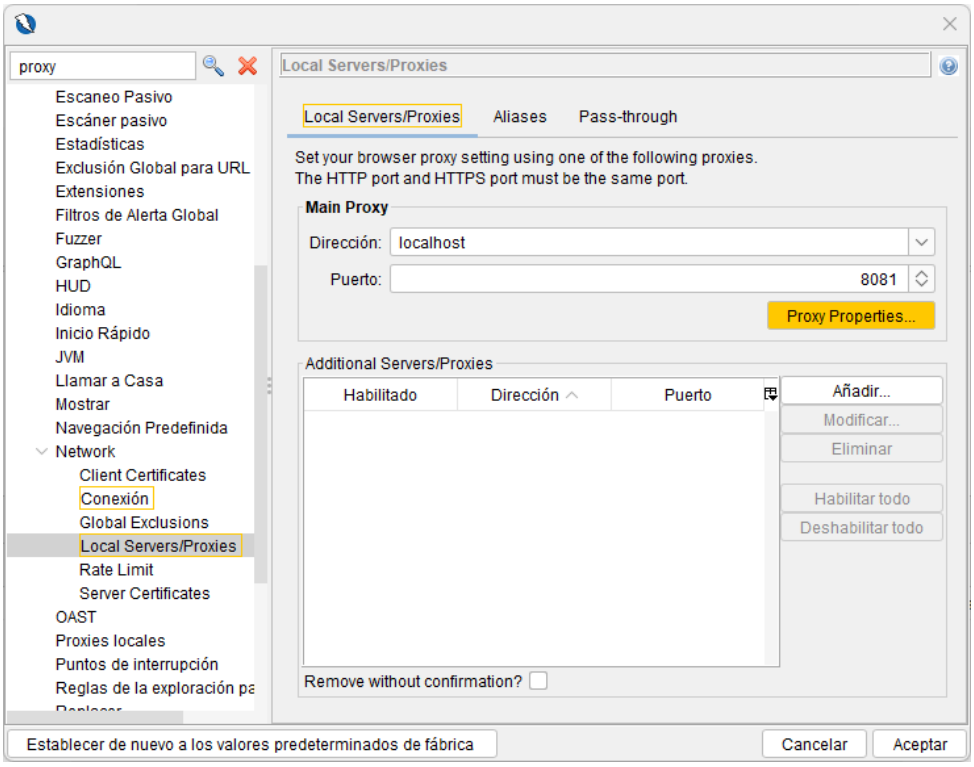


Figura 5: Opciones de proxy en opciones de la aplicación - Diseño del autor.

Se podrá observar la dirección como localhost (127.0.0.1) y el puerto como 8080 (en el caso que otra aplicación esté utilizando el puerto, podrá cambiarse, en este caso se cambió a 8180). En la parte inferior de la aplicación también se puede observar el proxy configurado (Figura 6).

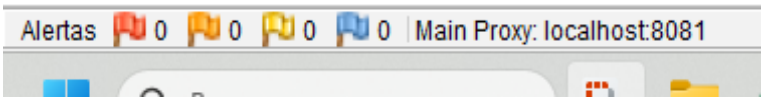


Figura 6: Proxy en la parte inferior de la aplicación - Diseño del autor.

2.2.1.4. Configurar el navegador para usar el Certificado Raíz:

Ya generado el certificado SSL dinámico, se debe instalar en la raíz del navegador web que será utilizado. Se puede hacer importando el certificado en la configuración de certificados del

navegador. En el caso de Mozilla Firefox, en las Preferencias del navegador, Privacidad y seguridad, en la parte de Certificados, “Ver certificados” (Figura 7) se podrán observar los certificados.

Figura 7: Configuración del navegador, certificados – Diseño del autor.

En la ventana de Administrar certificados, donde muestra la lista de certificados que contiene Firefox (Figura 8), se debe hacer clic en “Importar” para poder agregar el certificado creado desde ZAP.

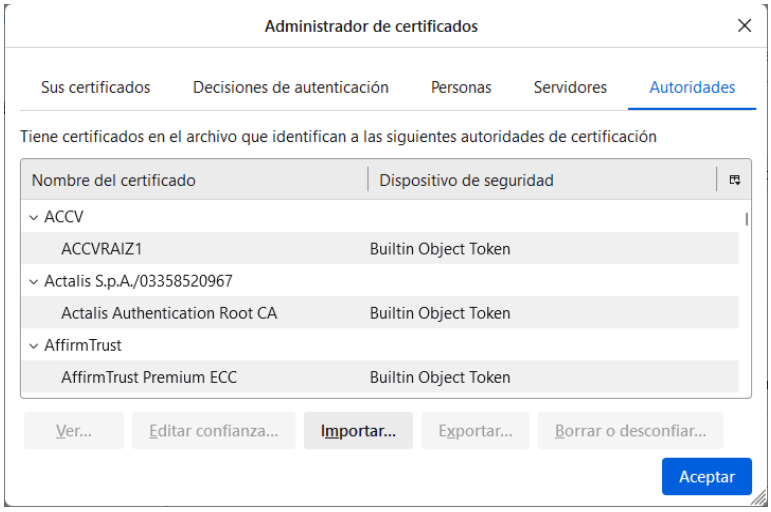


Figura 8: Administrador de certificados de Firefox - – Diseño del autor.

Una vez ya cargado el Certificado en el navegador, se debe configurar el proxy para que ZAP pueda comenzar a escanear. Para ello, también se debe cargar el proxy en el navegador. Al abrir el navegador, dirigiéndose a las “Opciones”, “Preferencias”, “General”, “Configuración de Red”. Se podrá visualizar la pantalla de Configuración de conexión, allí se selecciona en el radio button la Configuración manual del proxy, ingresando el mismo puerto que brinda la herramienta ZAP. Debe estar seleccionada la opción “También usar este proxy para HTTPS” para aumentar la seguridad al navegar por los sitios que se desean escanear (Figura 9).

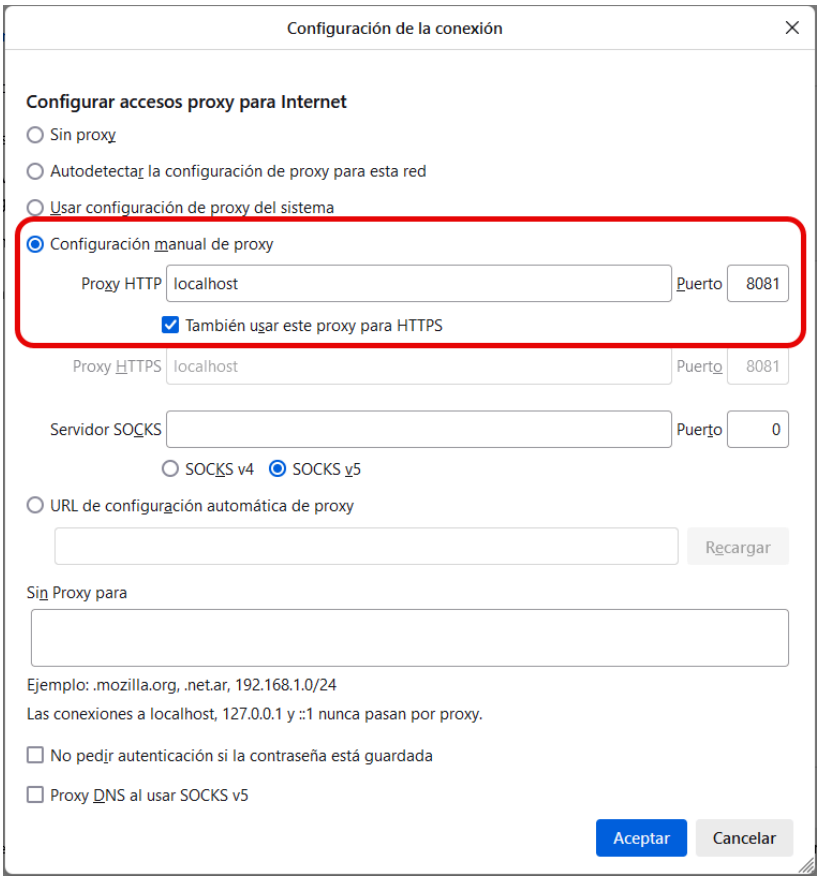


Figura 9: Ventana de configuración de conexión del navegador – Diseño del autor.

Luego de haber configurado el proxy, desde el navegador, ingresando a cualquier página web para verificar que, a través de la configuración del proxy generada, el programa comienza su escaneo del sitio.

En el navegador, si la conexión se realizó correctamente con la aplicación, se mostrará el mensaje de bienvenida a ZAP HUD con las herramientas a los costados izquierdo y derecho (Figura 10).

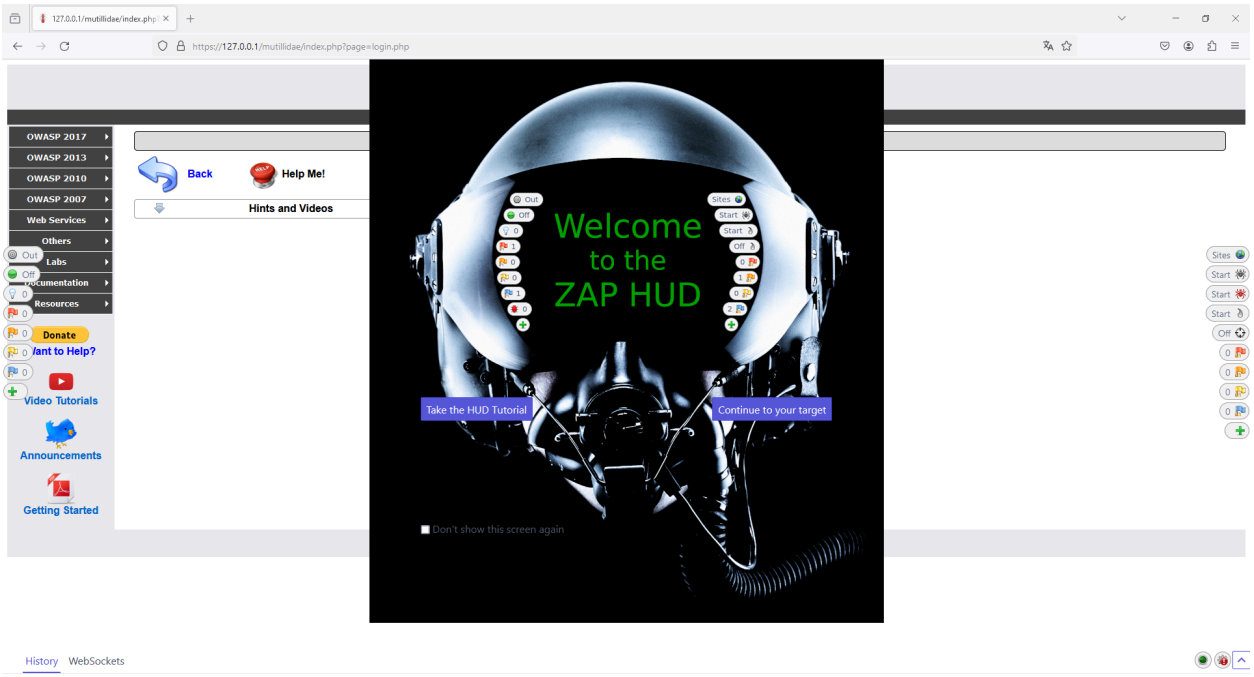


Figura 10: Mensaje de bienvenida de ZAP HUD con herramientas – Diseño del autor.

Se mostrará dos opciones para elegir, si desea realizar un tutorial o si se desea continuar y comenzar a navegar por el sitio. Al seleccionar la opción del tutorial, este brinda información sobre el HUD, "Heads Up Display" (Visualización Frontal), una característica que proporciona una vista rápida y conveniente de la información relevante sobre la seguridad mientras se navega por un sitio Web. El HUD se superpone en la pantalla del navegador web y muestra alertas de

seguridad en tiempo real, como advertencias de vulnerabilidad o información sobre las solicitudes HTTP interceptadas.

Algunas de las funciones del HUD son:

- **Alertas de Seguridad**: Muestra notificaciones instantáneas sobre posibles vulnerabilidades o riesgos de seguridad descubiertos por ZAP durante la navegación.
- **Información de Solicitudes y Respuestas**: Proporciona detalles sobre las solicitudes y respuestas HTTP interceptadas, lo que permite una comprensión más profunda del tráfico entre el navegador y el servidor.
- **Marcado de Páginas**: Permite marcar páginas específicas para un posterior análisis o revisión detallada.
- **Control de Intercepción**: Ofrece controles para habilitar o deshabilitar la interceptación de solicitudes y respuestas HTTP según sea necesario.

El HUD es una herramienta útil para realizar pruebas de penetración y evaluaciones de seguridad en aplicaciones web, ya que les permite mantenerse informados sobre posibles problemas de seguridad mientras interactúan con el sitio.

Además, el tutorial advierte, que se debe utilizar la herramienta con aplicaciones propias o con un permiso para realizar los ataques, ya que sin permiso puede constituir una violación de la ley y de las políticas de seguridad, además de ser éticamente cuestionable.

Luego de haber terminado con el tutorial, se debe ingresar a ZAP para corroborar que la conexión se estableció correctamente con el navegador (Figura 11).

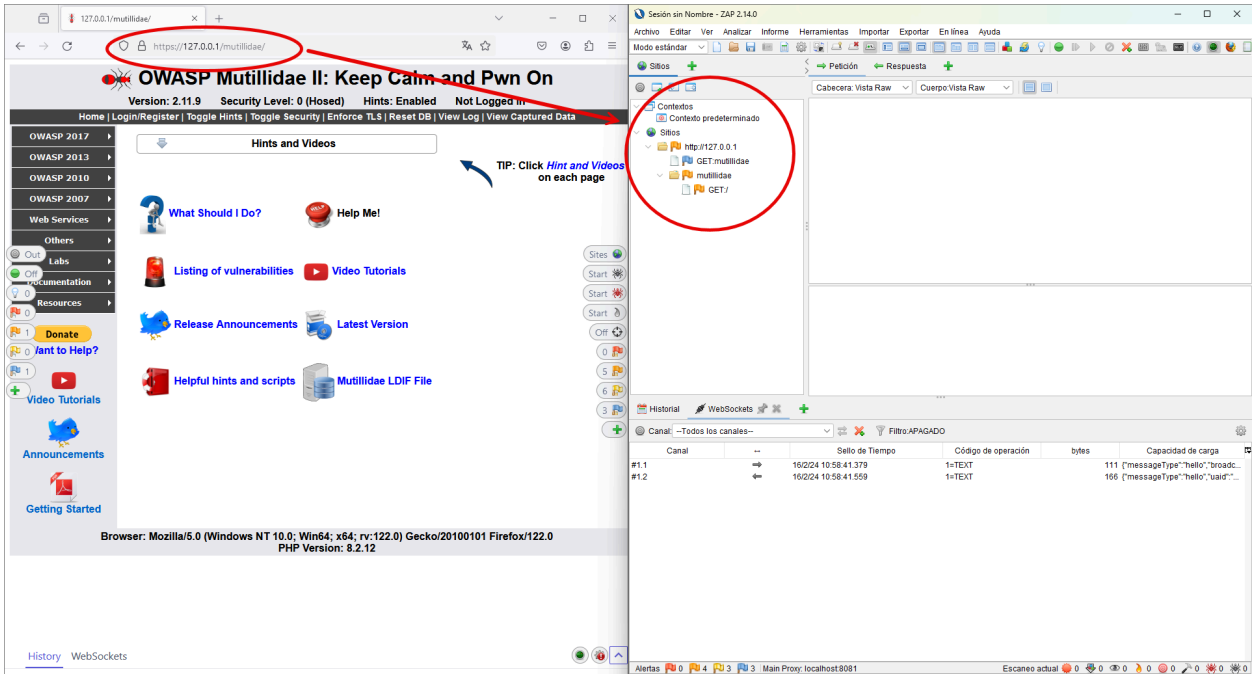


Figura 11: Conexión establecida de ZAP con el navegador – Diseño del autor.

En la imagen se observa que a través de un sitio que se encuentra alojado localmente, ZAP comienza a cargar las solicitudes brindadas en el sitio. Esto indica que el proxy cargado en el navegador está configurado correctamente.

Existen varias extensiones para cambiar proxis en los navegadores web. Estas extensiones suelen proporcionar una forma conveniente de cambiar entre diferentes proxis o configuraciones de proxy sin tener que modificar la configuración del navegador manualmente. Aquí hay algunas extensiones populares para diferentes navegadores:

- **Proxy SwitchyOmega** (Google Chrome): Esta es una extensión de Chrome que permite configurar múltiples perfiles de proxy y cambiar fácilmente entre ellos. Además, ofrece características avanzadas como la configuración de reglas de enrutamiento para determinados sitios web.

- **FoxyProxy Standard** (Mozilla Firefox): FoxyProxy es una extensión para Firefox que permite cambiar entre diferentes proxis de forma rápida y sencilla. Se pueden configurar múltiples proxis y asignarlos a diferentes sitios web o patrones de URL.
- **Proxy SwitchySharp** (Google Chrome): Similar a Proxy SwitchyOmega, esta extensión para Chrome permite gestionar múltiples perfiles de proxy y cambiar entre ellos con facilidad. También ofrece funcionalidades avanzadas como la configuración de reglas de enrutamiento.
- **SwitchyOmega** (Microsoft Edge): Una versión compatible con Microsoft Edge de la popular extensión Proxy SwitchyOmega para Chrome.
- **Proxy Helper** (Opera): Esta extensión para el navegador Opera permite cambiar entre diferentes proxis de forma sencilla y rápida.

2.2.1.5. Intercepción de solicitudes

Después de la configuración del proxy en el navegador, ingresando el sitio que se desea escanear, en ZAP comienzan a llegar las solicitudes generadas. El árbol comienza a construirse a medida que se realicen acciones en el sitio y en la Ventana de información, en la pestaña “Historial”, se puede observar el historial de las solicitudes en orden descendente para ver primero las últimas solicitudes. Dicho historial, está compuesto por el “Identificador de la solicitud”, la “Fuente”, la “Petición”, donde muestra la fecha y la hora de la petición, el método utilizado, la URL, los códigos de estado, el RTT "Round-Trip Time"[4], el tamaño de la respuesta, los alertas, entre otros. Esta tabla brinda la información necesaria de las solicitudes para poder evaluarlas (Figura 12).

[4] RTT: (Tiempo de ida y vuelta) se refiere al tiempo que tarda un paquete de datos en viajar desde un origen a un destino y luego regresar al origen.

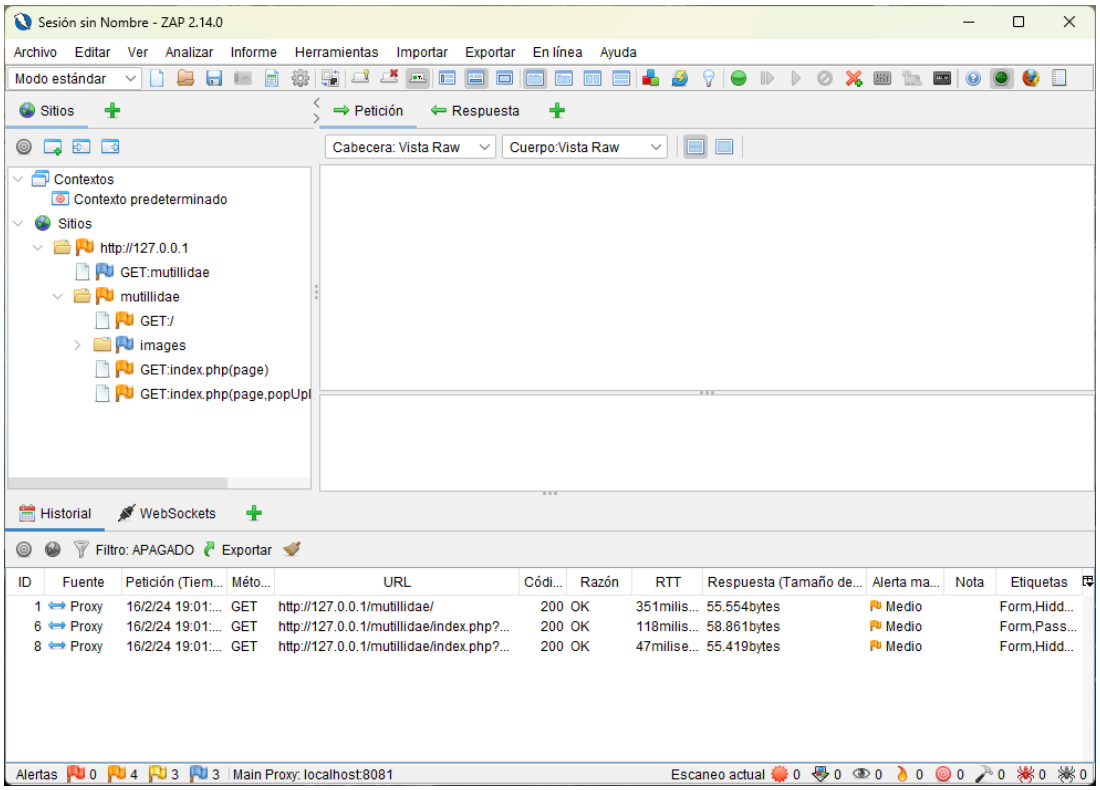


Figura 12: Solicitudes generadas al interceptar solicitudes - Diseño del autor.

Ya que la conexión se estableció correctamente entre el sitio Web y la herramienta, se podrá comenzar a interceptar solicitudes con las acciones que se realizan al navegar por el sitio.

Para comenzar a analizar el sitio, se accede a la sección de inicio de sesión, se procede a ingresar los datos, en este caso: *Usuario: admin, contraseña: adminpass* (Figura 13).

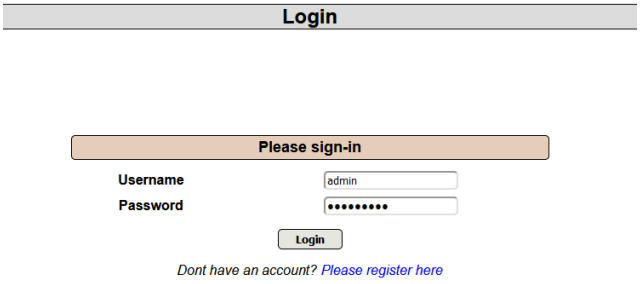


Figura 13: Ingreso de datos en el navegador – Diseño del autor.

Al hacer clic en Login, ZAP comienza a recibir las solicitudes y se registran en la barra de historial de la aplicación, donde se observa la solicitud de inicio de sesión o login (Figura 14).

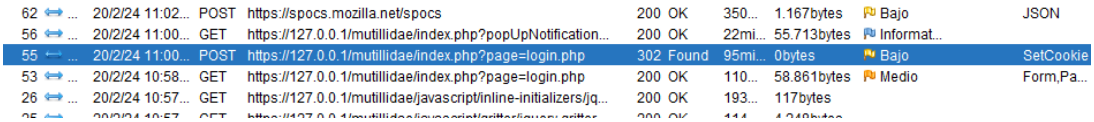


Figura 14: Solicitud de Login en historial de solicitudes – Diseño del autor.

Al seleccionar la solicitud de *login*, en el Espacio de trabajo, en la pestaña Request (Solicitudes) se puede observar la información de la solicitud; el cuerpo de la respuesta, los datos que se ingresaron anteriormente y la respuesta (Figura 15).

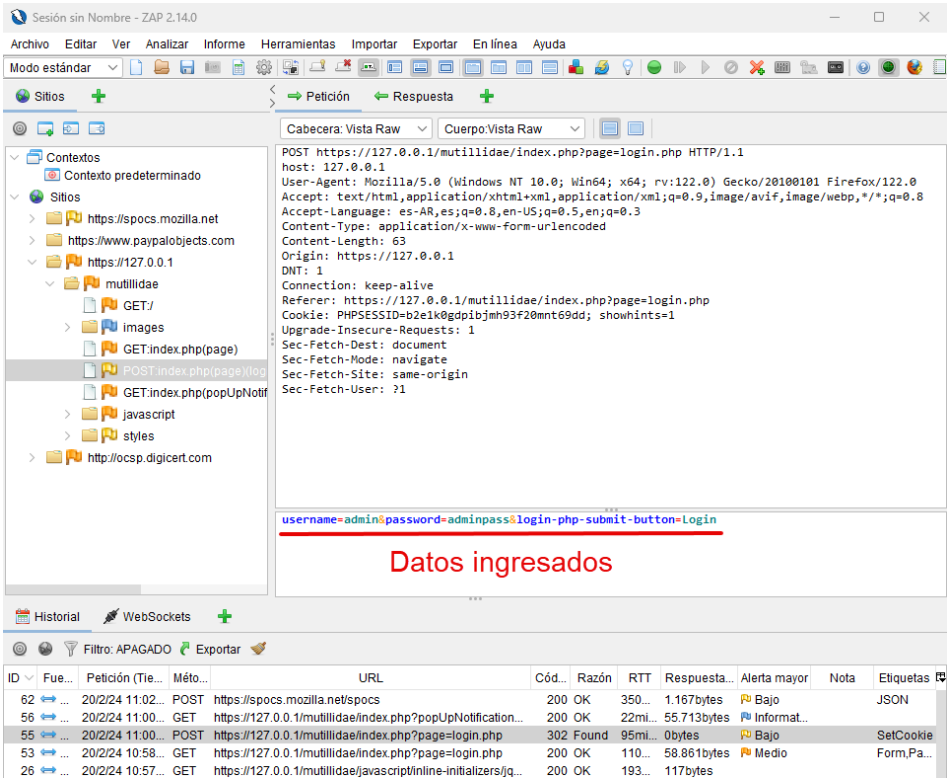


Figura 15: Registro de datos ingresados en el navegador – Diseño del autor.

De esta manera se podrá interceptar las solicitudes con su respuesta, lo que resulta de gran utilidad poder ver las solicitudes generadas con dichas respuestas y así poder evaluarlas.

2.2.1.6. Puntos de interrupción

Cuando la conexión del sitio web con la aplicación se establece correctamente, ZAP contiene la opción de crear Puntos de interrupción. La aplicación capta las solicitudes del navegador y no envía las Respuestas hasta que ZAP lo permita (Figura 16)

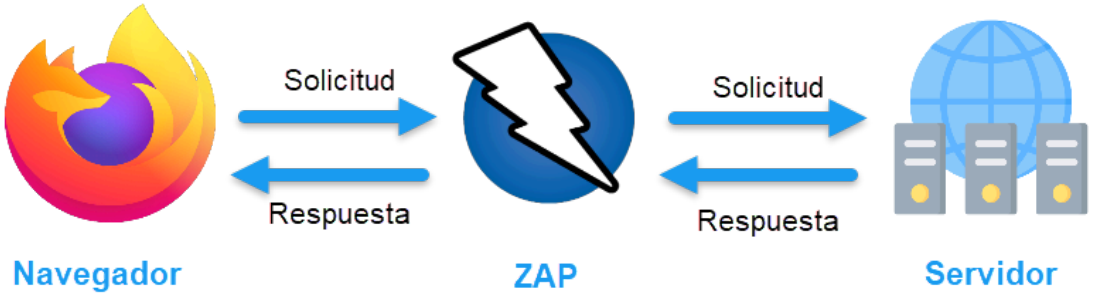


Figura 16: Intercepción de solicitudes y respuestas en ZAP - Diseño del autor.

Para activar los puntos de interrupción, se debe hacer clic en el círculo verde que se encuentra en la barra de herramientas. Al activar el botón se convierte en rojo, esto indica que se interceptó un punto (Figura 17).

Figura 17: Cambio de estado del botón de Punto de interrupción - Diseño del autor.

Una vez activado el punto de interrupción, se debe dirigir al navegador e intentar realizar una acción, en este caso iniciar sesión en el sitio.

Se observa que al hacer clic en “Login” para poder iniciar sesión, ZAP recibió la solicitud con la información con los datos ingresados mientras el navegador espera la respuesta para proceder (Figura 18).

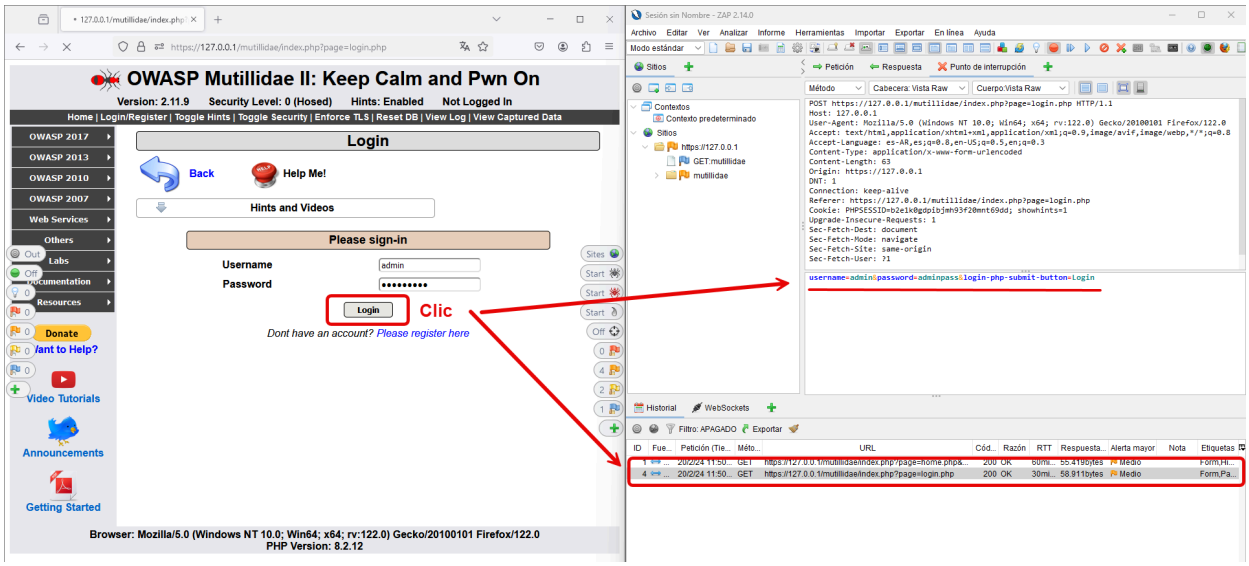


Figura 18: Datos ingresados en el navegador con la solicitud - Diseño del autor.

A continuación, para poder enviar la respuesta, se debe hacer clic el en icono del triángulo de color azul en la barra de herramientas (Figura 19).

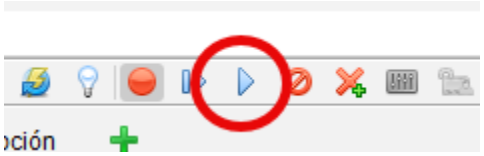


Figura 19: Botón de inicio de interrupción en barra de herramientas - Diseño del autor.

Luego de reanudar la interrupción, se puede observar la respuesta en el navegador, en este caso el inicio de sesión exitoso (Figura 20).

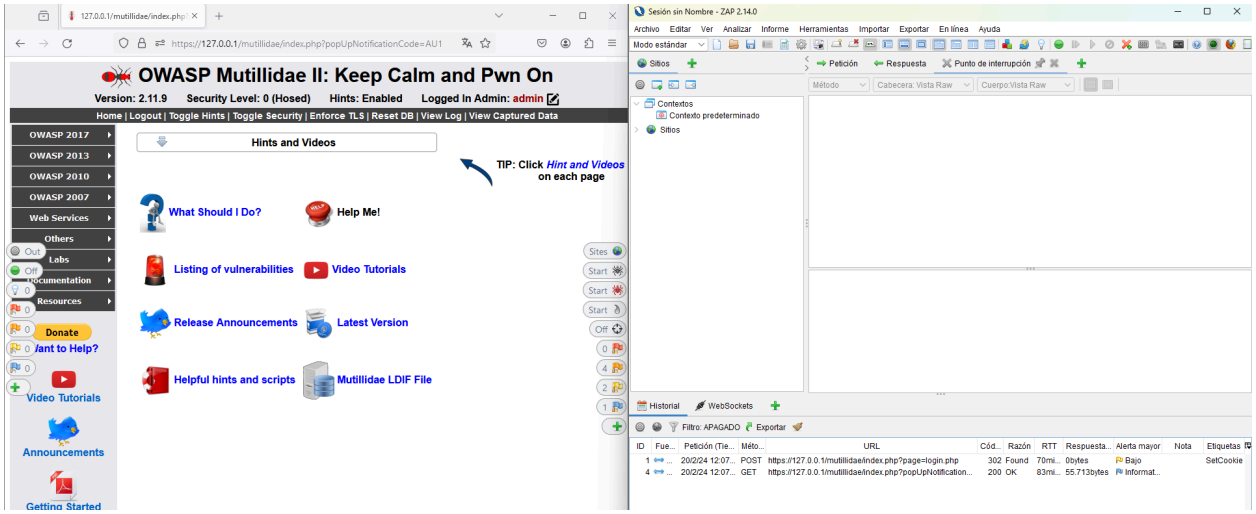


Figura 20: Reanudación de interrpción en el navegador – Diseño del autor.

Los puntos de interrupción, también permiten modificar la información de las solicitudes recibidas por parte del navegador para poder realizar testeos. En este caso, al modificar los datos de la contraseña "*password=adminpass333*" (dato incorrecto) en ZAP (Figura 21), se envía la respuesta y se observa que no se pudo continuar con el inicio de sesión en el sitio por los datos incorrectos (Figura 22).

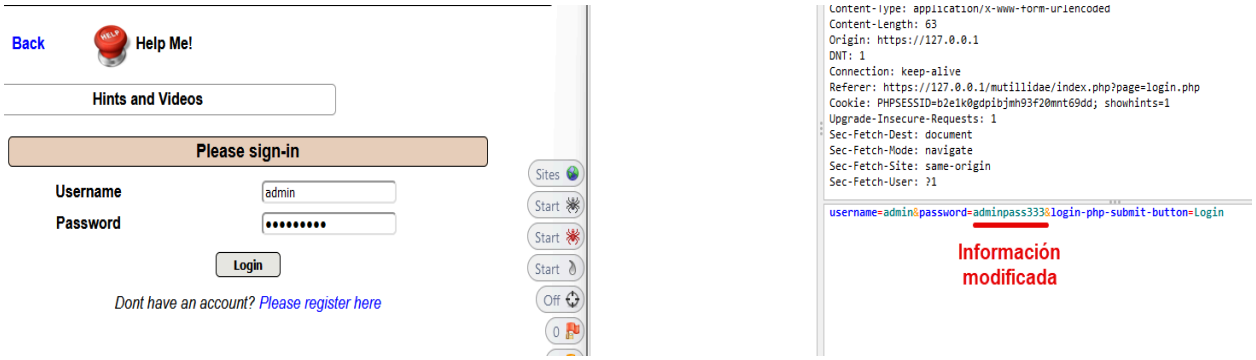


Figura 21: Modificación de datos ingresados en ZAP – Diseño del autor.

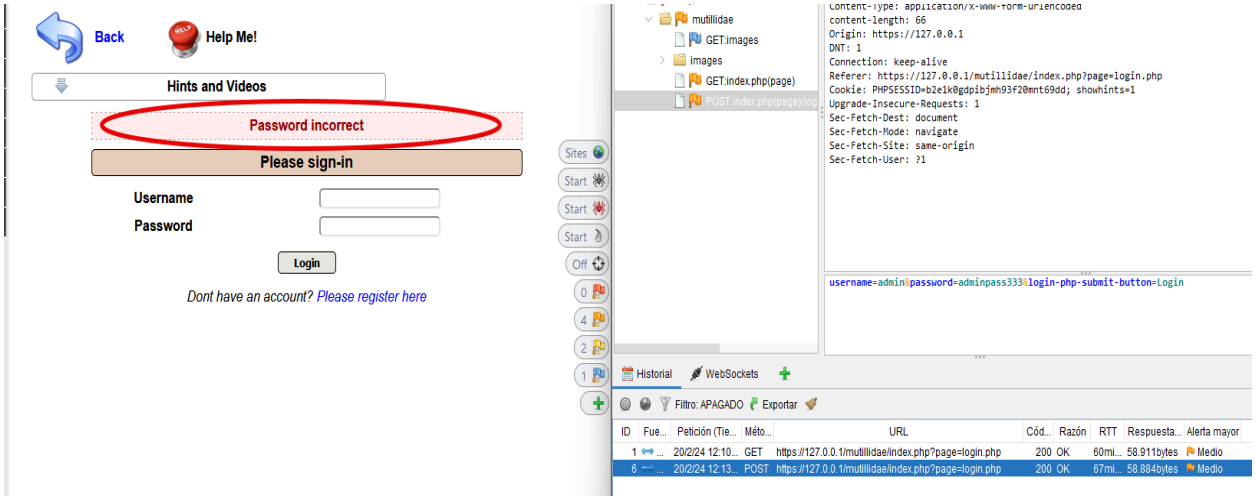


Figura 22: Respuesta en el navegador con datos incorrectos – Diseño del autor.

De este modo, se puede realizar testeos interceptando las solicitudes para luego poder evaluarlas.

2.2.1.7. Interceptando una solicitud Específica

Al navegar por las diferentes páginas del sitio web, comenzará a crear la vista de árbol, además se registrará en el historial de la ventana de información. En el caso que, si se desea interceptar una sección en particular, por ejemplo, en la sección de búsqueda DNS, es posible capturar esa sección cada vez que se visita esa página, se debe hacer clic con el botón secundario del ratón en la solicitud y se selecciona la opción “Interrupción” (Figura 23).

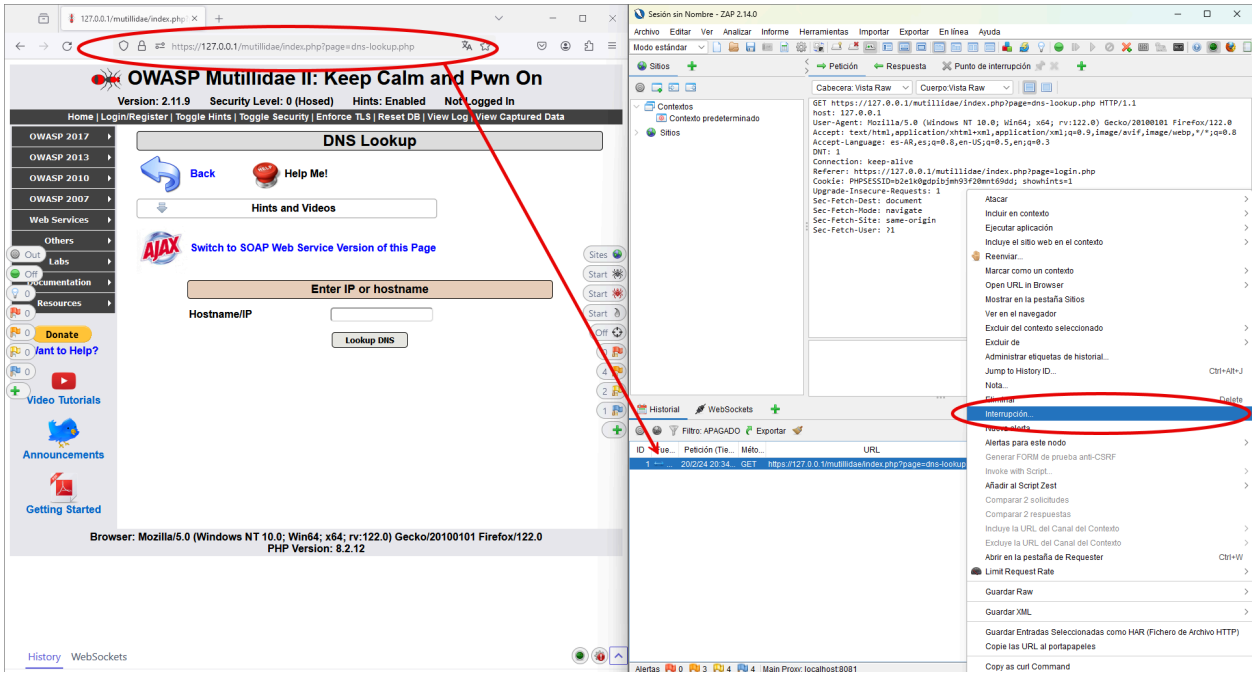


Figura 23: Interrupción de solicitudes – Diseño del autor.

Al realizar la interrupción se mostrará una ventana con información como la ubicación, la coincidencia y la Cadena; que en este caso es la URL del sitio (Figura 24).

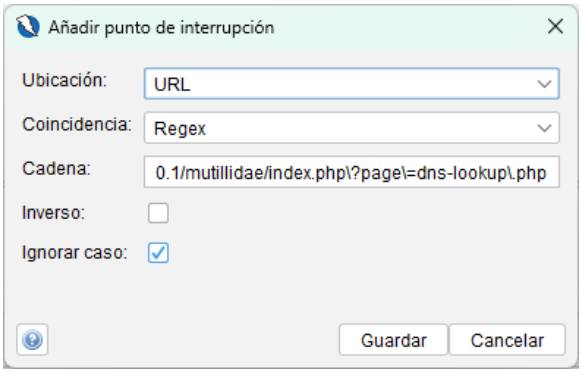


Figura 24: Ventana de para añadir un punto de interrupción – Diseño del autor.

Haciendo clic en "Guardar", la aplicación capta esa solicitud que luego si se ingresa a esa sección lo cual se creará el punto. Se observan todos los puntos de interrupción agregados en la pestaña de "Puntos de interrupción" en la ventana de información de la aplicación (Figura 25).

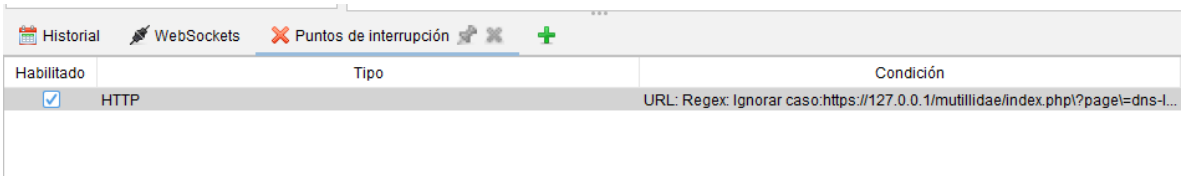


Figura 25: Solicitud en pestaña de punto de interrupción – Diseño del autor.

Al hacer clic en el navegador donde está creado el punto, se habilitarán las opciones de interrupción en la barra de herramientas (Figura 26). De este modo se podrán interceptar secciones de la web específicas.

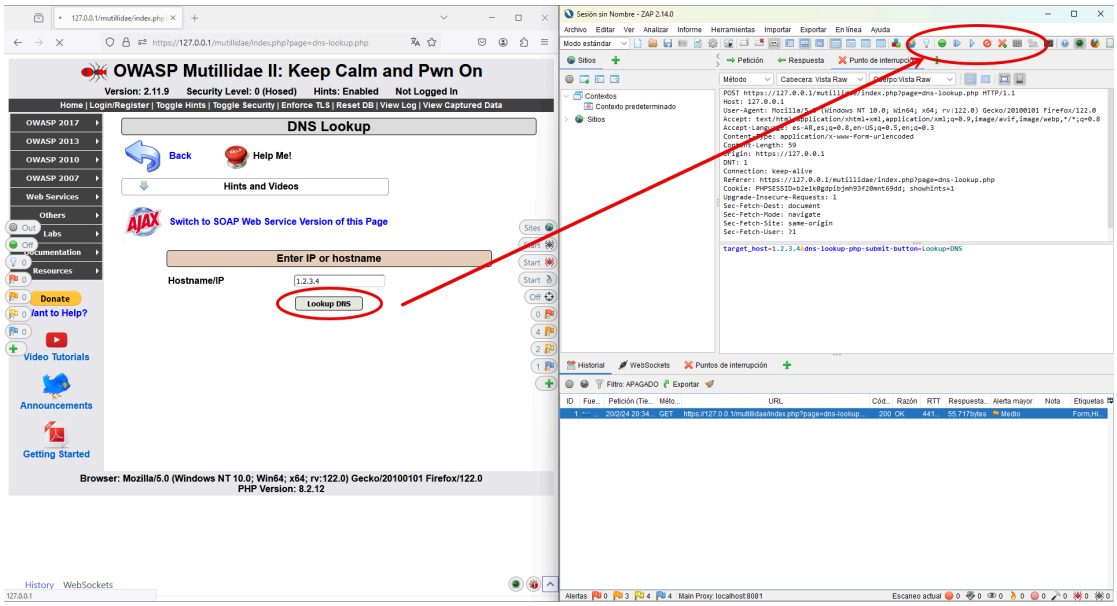


Figura 26: Opciones de interrupción habilitadas – Diseño del autor.

2.2.1.8. Exploración Manual

Al iniciar ZAP, se podrán observar principalmente las opciones de Exploración Manual en el Espacio de trabajo, donde muestra un mensaje de bienvenida y una breve descripción de la herramienta (Figura 27).

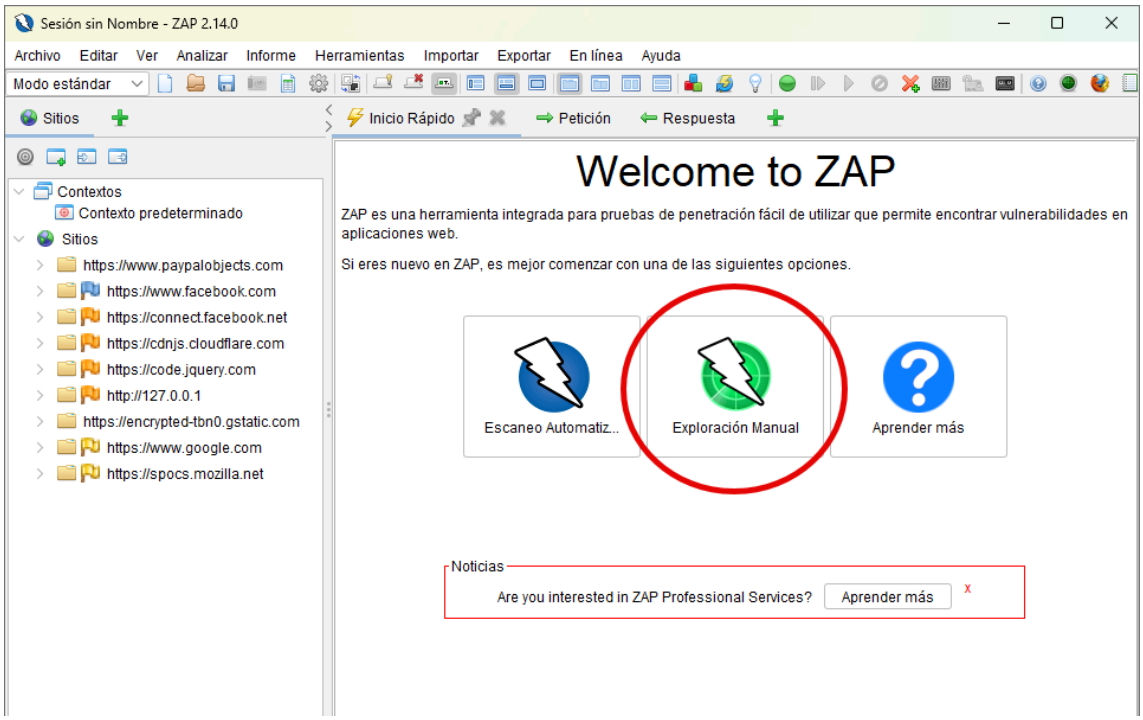


Figura 27: Exploración manual en espacio de trabajo – Diseño del autor

Aquí se solicitará que se ingrese la URL del sitio que se desea escanear (Figura 28). Además, se deberá elegir el navegador para comenzar el escaneo, los cuales pueden ser: Firefox o Chrome (en este caso se selecciona Firefox). Al hacer clic en el botón “Iniciar Navegador” se abrirá el navegador seleccionado con la URL que se ingresó anteriormente. Luego se observa que en la aplicación comienza a construirse el árbol y el historial a medida que realiza la carga del sitio (Figura 29).

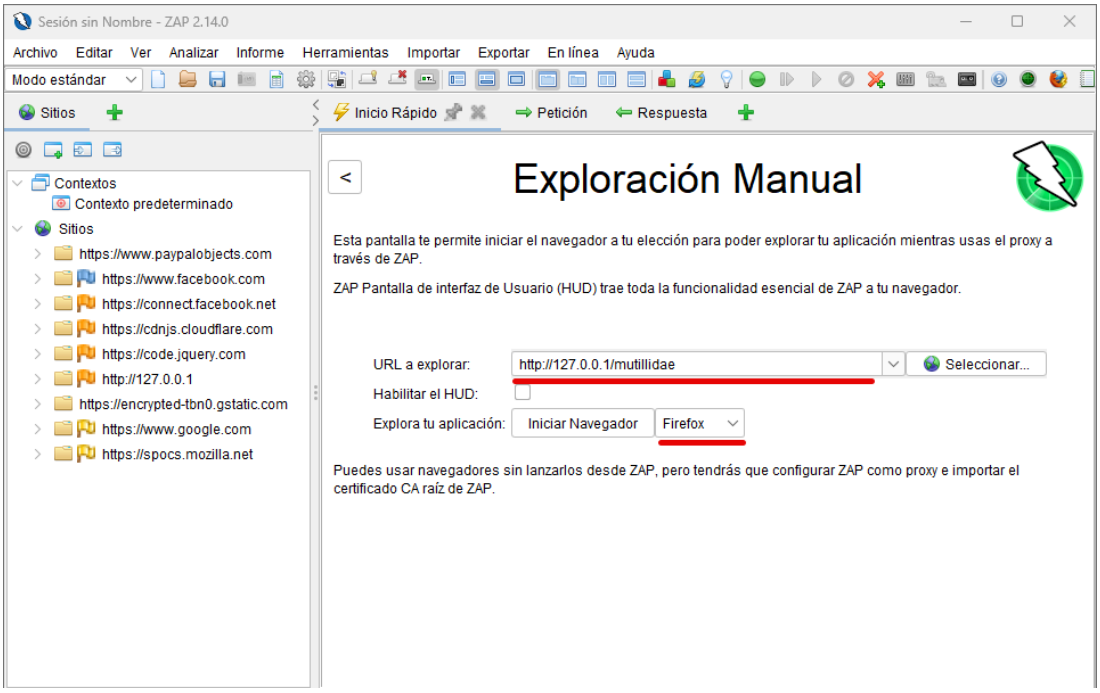


Figura 28: Ingreso de URL en Exploración manual – Diseño del autor.

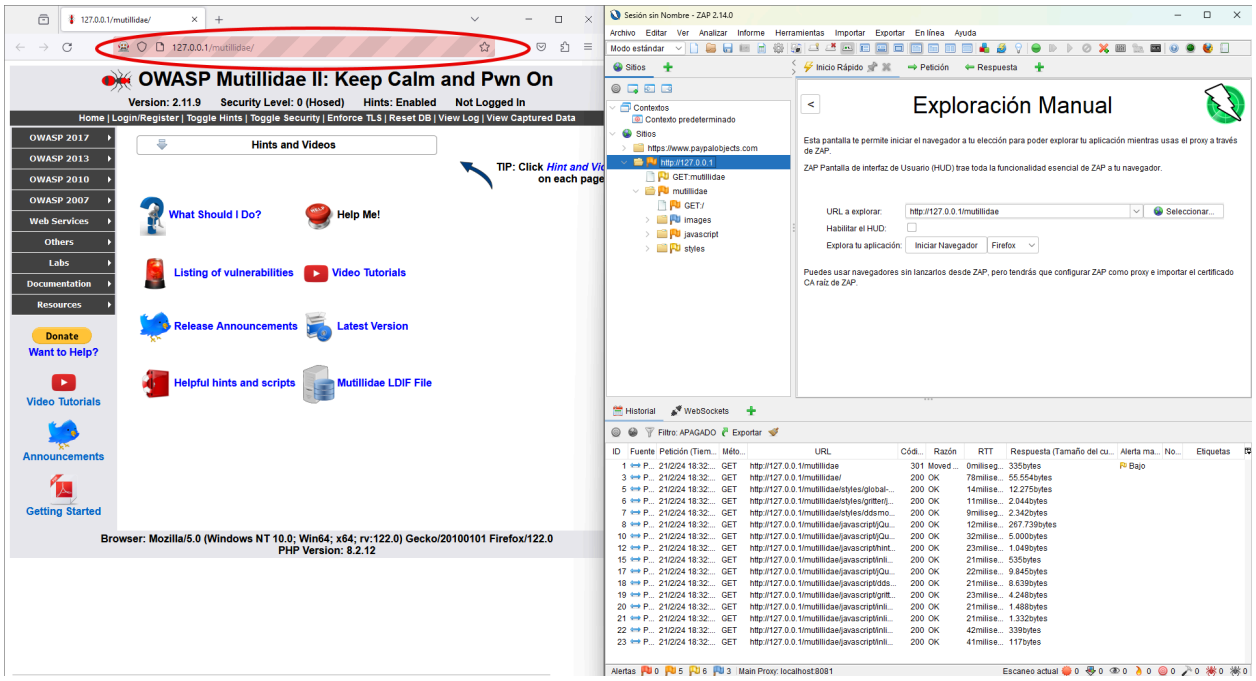


Figura 29: Carga del sitio en el navegador – Diseño del autor.

Al utilizar esta opción de Exploración Manual, el navegador ya estará configurado, es decir que no será necesario cargar un certificado SSR, ni cambiar el proxy. Está opción es muy útil cuando se está detrás de un proxy corporativo y no se puede cambiar el proxy desde el navegador externo.

2.2.1.9. Escaneos Automatizados

Un escaneo automatizado en OWASP ZAP, es un proceso en el cual la herramienta realiza pruebas de forma automatizada, sin intervención manual significativa por parte del usuario. Durante este proceso, ZAP utiliza diversas técnicas para identificar posibles vulnerabilidades y riesgos de seguridad en la aplicación web objetivo.

El escaneo automatizado en ZAP implica varias etapas, que pueden incluir:

- **Spidering (Rastreo):** ZAP utiliza un spider (araña) para recorrer y mapear la estructura de la aplicación web, descubriendo enlaces y páginas.
- **Escaneo Activo y Pasivo**: ZAP envía solicitudes modificadas y analiza las respuestas para detectar posibles vulnerabilidades. El escaneo puede ser activo, donde ZAP interactúa directamente con la aplicación enviando solicitudes específicas diseñadas para detectar vulnerabilidades, o pasivo, donde ZAP simplemente observa el tráfico sin interactuar con la aplicación.
- **Análisis de Resultados**: ZAP analiza las respuestas recibidas y las compara con patrones y firmas conocidas de vulnerabilidades. Si se encuentra una posible vulnerabilidad, ZAP genera una alerta para notificar al usuario.

- **Reporte de Hallazgos**: Una vez completado el escaneo, ZAP proporciona un informe detallado de los hallazgos, que incluye una lista de posibles vulnerabilidades encontradas, su gravedad, y recomendaciones para mitigar o corregir los problemas detectados.

2.2.1.10. Spidering

La Exploración Manual es muy efectiva para encontrar vulnerabilidades en una aplicación cuando conoce el flujo de toda la aplicación para realizar el análisis. El problema es que se necesita de una persona para realizar el escaneo y en algunos se le pueden pasar algunas vulnerabilidades.

En OWASP ZAP, las "spiders" (arañas) se refieren a una función de escaneo automático que recorre y descubre enlaces y páginas dentro de un sitio web objetivo. Esta función es parte del proceso de "crawling" (rastreo), que es el primer paso en el escaneo de seguridad de una aplicación web.

El spider de ZAP simula el comportamiento de un navegador web al seguir los enlaces encontrados en una página inicial y luego en las páginas subsiguientes. Esto le permite mapear la estructura del sitio web y descubrir posibles puntos de entrada para las pruebas de seguridad.

Algunas de las acciones que realiza el spider de ZAP incluyen:

- **Seguir enlaces**: El spider sigue los enlaces encontrados en una página para descubrir nuevas páginas y recursos dentro del sitio web.
- **Mapear la estructura del sitio**: El spider construye un mapa del sitio web, donde se mostrarán cómo están interconectadas las diferentes páginas y secciones.
- **Descubrir endpoints**: Identifica endpoints y puntos de acceso a la aplicación, que luego pueden ser objeto de pruebas de seguridad más exhaustivas.

- **Recopilar datos**: Durante el proceso de spidering, ZAP puede recopilar información sobre los parámetros de las solicitudes HTTP y otras características relevantes de las páginas visitadas.

El spider es una herramienta útil para automatizar la identificación y el mapeo de sitios web durante las pruebas de seguridad, lo que ayuda a los profesionales de seguridad a comprender mejor la superficie de ataque y a priorizar sus esfuerzos de pruebas.

Hay varias formas de iniciar Spiders, una de las formas es hacer clic en el icono “más” en las opciones de la Ventana de información seleccionado “Spider (Araña)” (Figura 30).

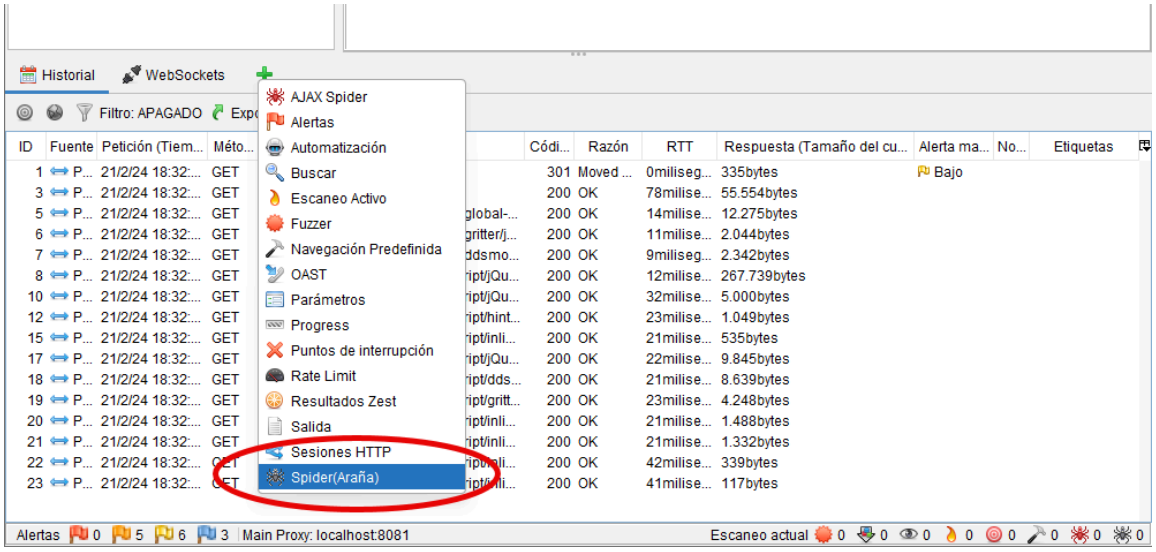


Figura 30: Inicio de Spider – Diseño del autor.

Se habilitarán nuevas opciones de Spider. En estas opciones se encontrará “Nuevo escaneo”, al hacer clic, abrirá una ventana de diálogo, en ella, se debe ingresar el Punto Inicial, en este caso, se ingresa la URL del sitio para rastrear el contexto de los parámetros y se inicia el escaneo (Figura 31).

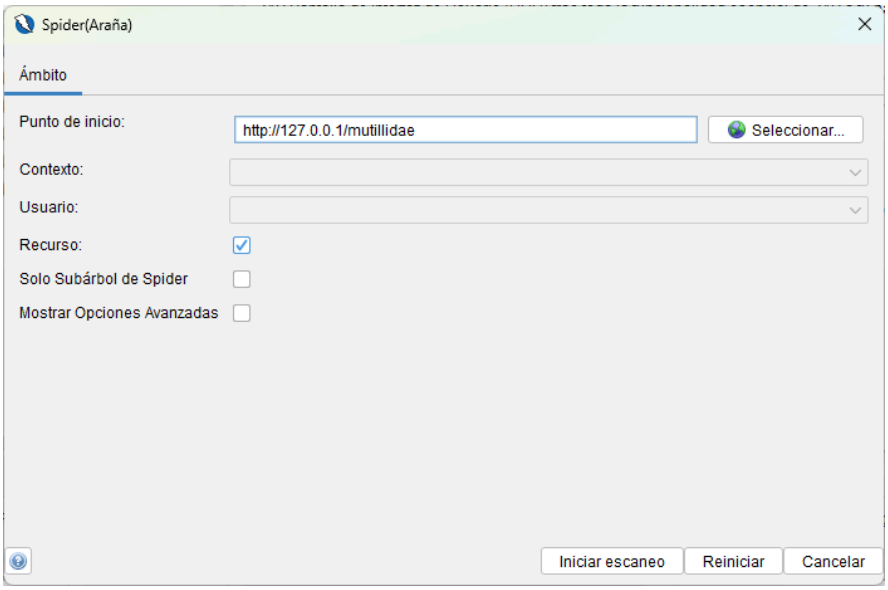


Figura 31: Ventana de Spider – Diseño del autor.

A medida que avanza el escaneo, se puede observar que el árbol del sitio comienza a construirse y en la pestaña de Spider se puede ver el progreso del escaneo actual y al finalizar indicará cuántas URL encontró, enumerándolas (Figura 32).

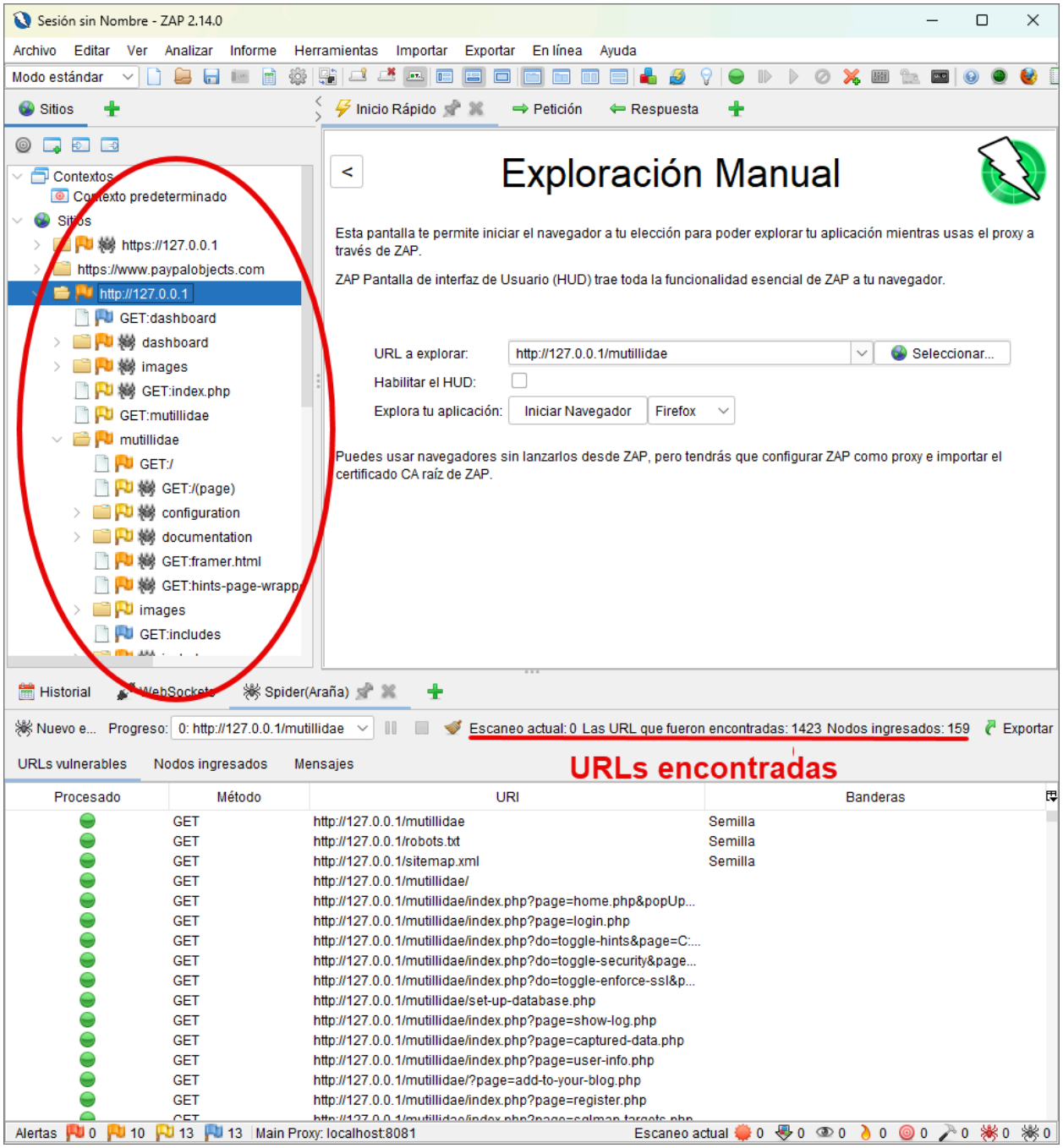


Figura 32: Construcción de Árbol y progreso de escaneo de Spider – Diseño del autor.

Además de las URL, se observan los Nodos ingresados y los mensajes que se enviaron por lo que se podrá ver los detalles de estos mensajes, con las respuestas (Figura 33 y 34).

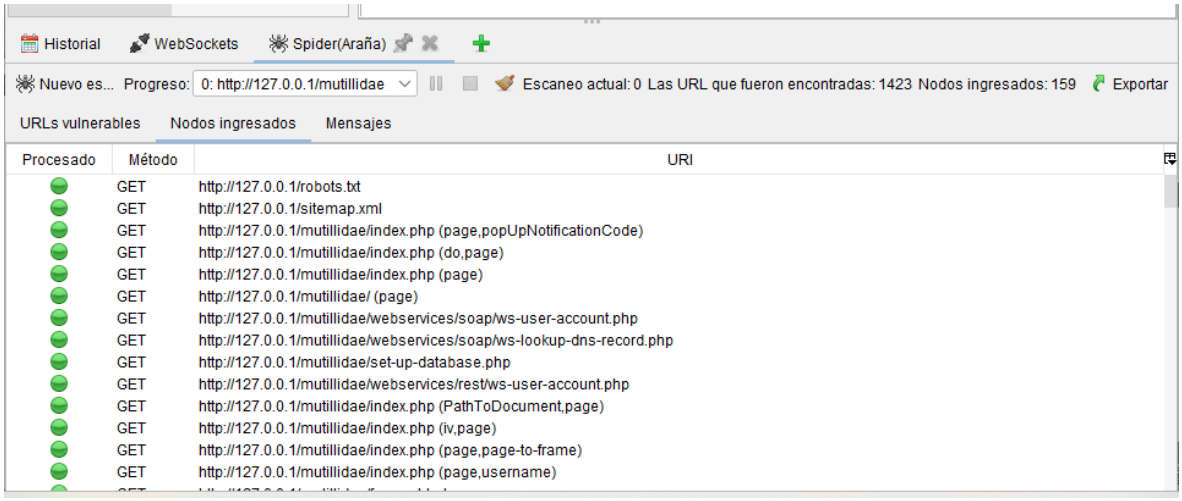


Figura 33: Lista de nodos ingresados – Diseño del autor.

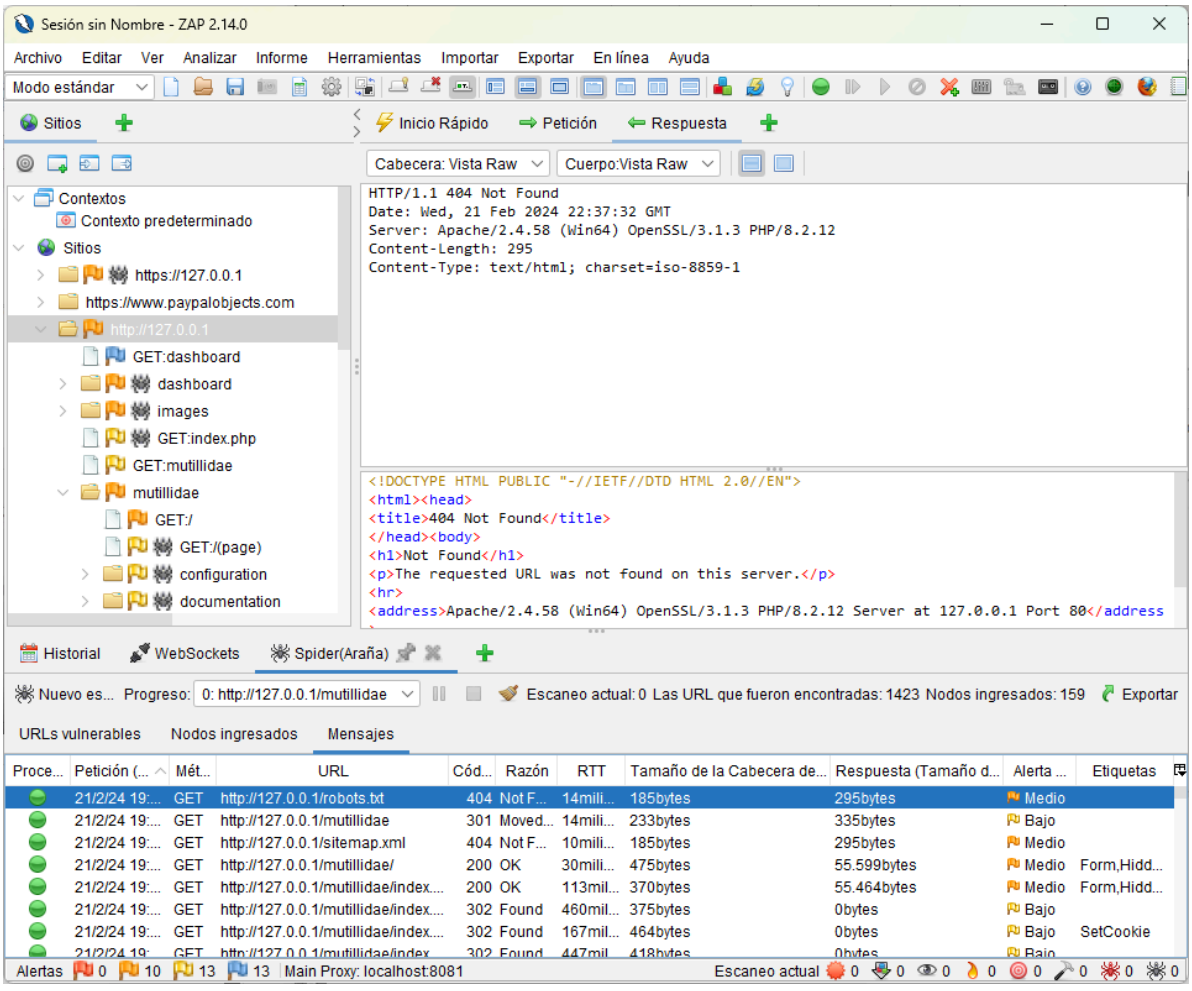


Figura 34: Mensajes enviados – Diseño del autor.

Los "Nodos" en el spider se refieren a los distintos elementos que el spider descubre y mapea durante su recorrido por un sitio web objetivo. Cada nodo representa una página web individual, un recurso (como una imagen, un archivo CSS o JavaScript), o cualquier otro elemento que el spider haya encontrado mientras rastrea el sitio.

Los nodos son esenciales para comprender la estructura y el contenido del sitio web analizado. El spider utiliza estos nodos para construir un mapa del sitio, mostrando cómo están interconectadas las diferentes páginas y secciones.

Algunas características importantes de los nodos en el spider de OWASP ZAP incluyen:

1. **URL**: Cada nodo tiene una URL única que identifica la ubicación del recurso dentro del sitio web.
2. **Tipo de Recurso**: Los nodos pueden representar diferentes tipos de recursos, como páginas HTML, imágenes, archivos CSS, JavaScript, entre otros.
3. **Relaciones**: Los nodos pueden estar interconectados a través de enlaces dentro del sitio web. El spider utiliza estos enlaces para navegar de un nodo a otro y mapear la estructura del sitio.
4. **Parámetros y Datos**: Algunos nodos pueden contener parámetros o datos relevantes, como formularios, cookies o parámetros de URL, que pueden ser importantes para el análisis de seguridad.

Comprender la estructura de los nodos descubiertos por el spider es fundamental para identificar posibles puntos de entrada para las pruebas de seguridad y para comprender la superficie de ataque de un sitio web.

2.2.1.11. Escaneo Activo

Un escaneo activo en ZAP es un proceso automatizado mediante el cual la herramienta envía solicitudes y analiza las respuestas de una aplicación web con el fin de identificar posibles vulnerabilidades de seguridad.

Durante un escaneo activo, ZAP puede llevar a cabo una variedad de acciones, como enviar solicitudes HTTP modificadas, realizar inyecciones de código, intentar la enumeración de recursos, entre otros. Estas acciones se realizan de manera automatizada y controlada, con el objetivo de identificar vulnerabilidades como inyecciones SQL, ataques de scripting entre sitios (XSS)[5], inyecciones de comandos, entre otros.

Para realizar un ataque a un sitio y poder buscar las vulnerabilidades, solo se debe visitar el sitio que se desea, luego ZAP comienza con la carga del Árbol y del Historial del sitio. Solo queda comenzar a navegar por el sitio y dirigirse a la parte que se desea realizar el ataque. Una vez que se tiene identificado la URL en el Historial, se hace clic con el botón secundario del mouse, y en la lista hacer clic en la opción "Mostrar en la pestaña del Sitio" lo que se marcará en el árbol (Figura 35).

[5] XSS: Cross-Site Scripting, son un tipo de vulnerabilidad de seguridad que permite a un atacante inyectar scripts maliciosos en páginas web vistas por otros usuarios.

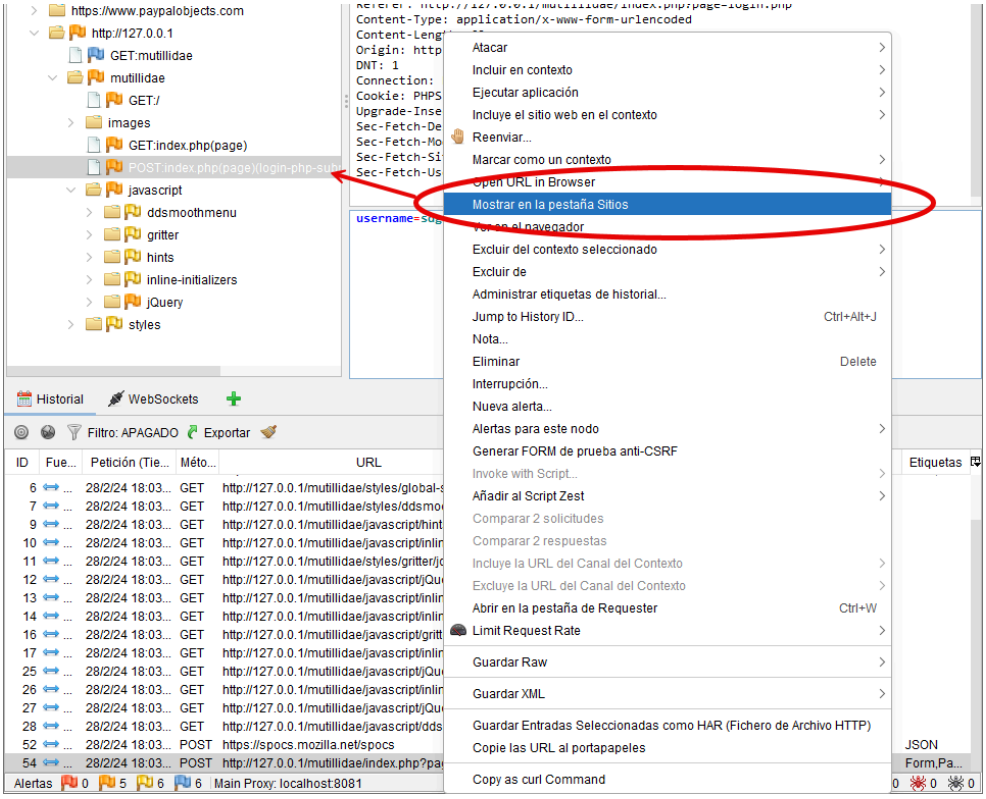


Figura 35: Opción mostrar en las pestaña los sitios – Diseño del autor.

Cuando se tiene identificada la URL en el árbol, se debe hacer clic con el botón secundario y seleccionar la opción de “Atacar”, luego en “Escaneo Activo” (Figura 36).

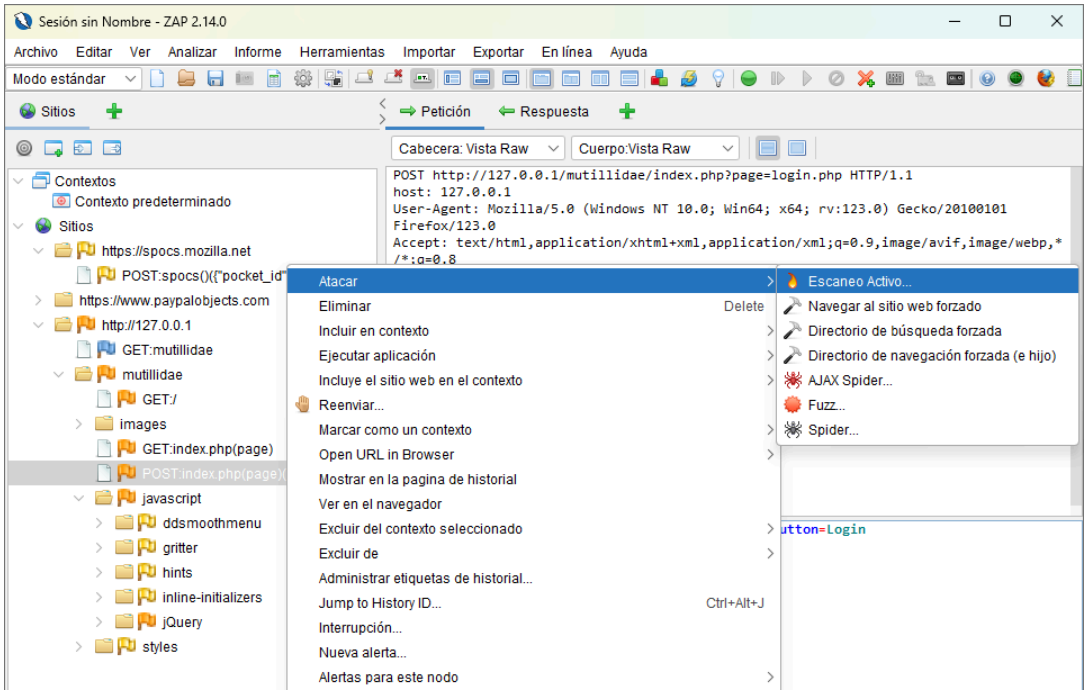


Figura 36: Opción de Escaneo Activo en Atacar – Diseño del autor.

Se abrirá una ventana, donde muestra el Punto de inicio que es la URL del sitio seleccionado y la Política por defecto. Para comenzar se debe hacer clic en “Comenzar Escaneo” (Figura 37).

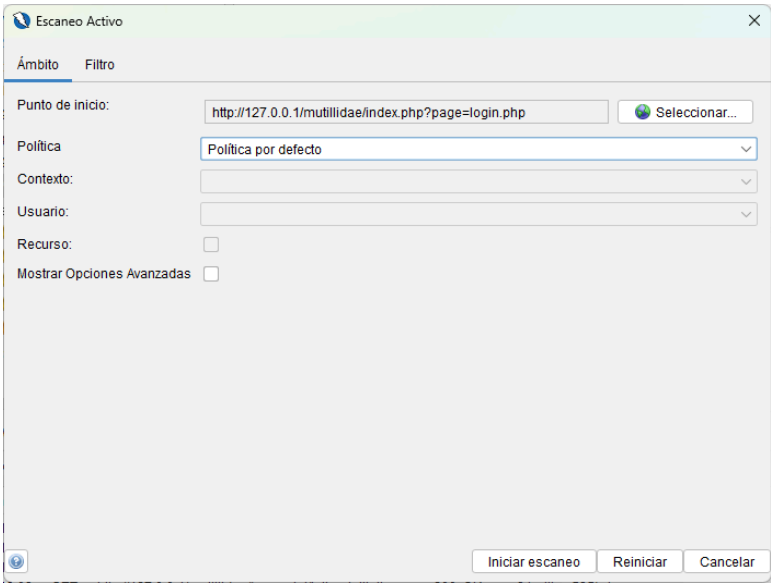


Figura 37: Ventana de opciones de Escaneo Activo – Diseño del autor.

En la Ventana de Información, se activará una nueva pestaña llamada “Escaneo Activo”, donde se verá el progreso del escaneo. A través de una barra muestra el progreso, además de dos botones para poder pausar y detener el escaneo. Una vez completado el análisis, mostrará los mensajes, la cantidad de Alertas de los problemas encontrados y el Número de peticiones realizadas (Figura 38).

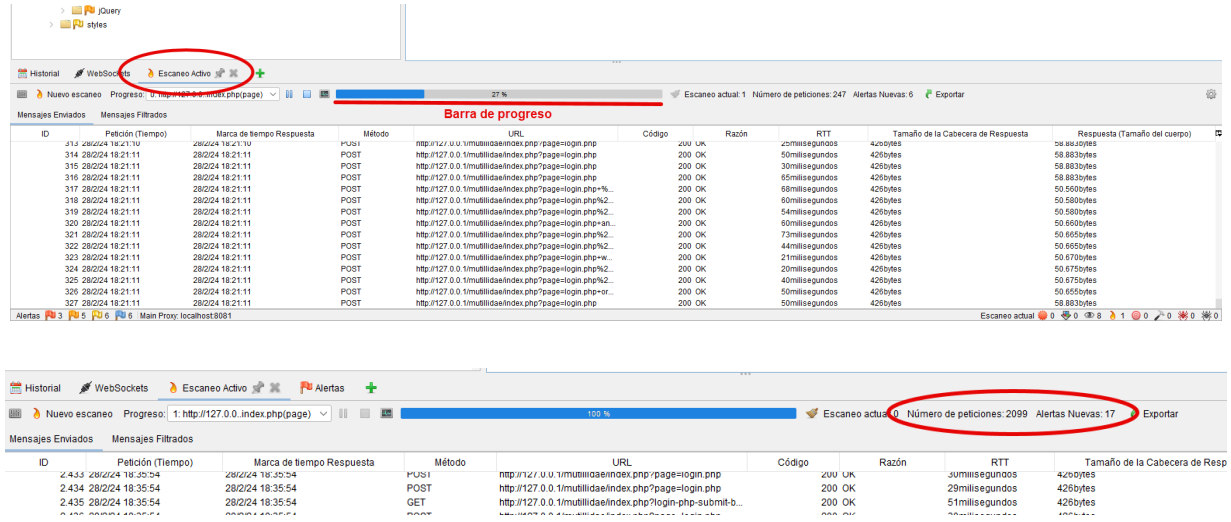


Figura 38: Pestaña de escaneo activo – Diseño del autor.

Luego de completar un Escaneo Activo en el sitio o en algún sector del mismo. Se habilitará una pestaña en la Ventana de Información llamada “Alertas”. En ese lugar se encontrará una lista con todos los Alertas encontrados en el sitio, ordenados por el nivel de Riego, del más alto al más bajo. Estas alertas están identificadas por una bandera y el color de la misma identifica el nivel del riesgo; rojo de alto riesgo, naranja y amarillo para medio y azul para el más bajo. Esto ayuda a poner énfasis en los más altos (Figura 39).

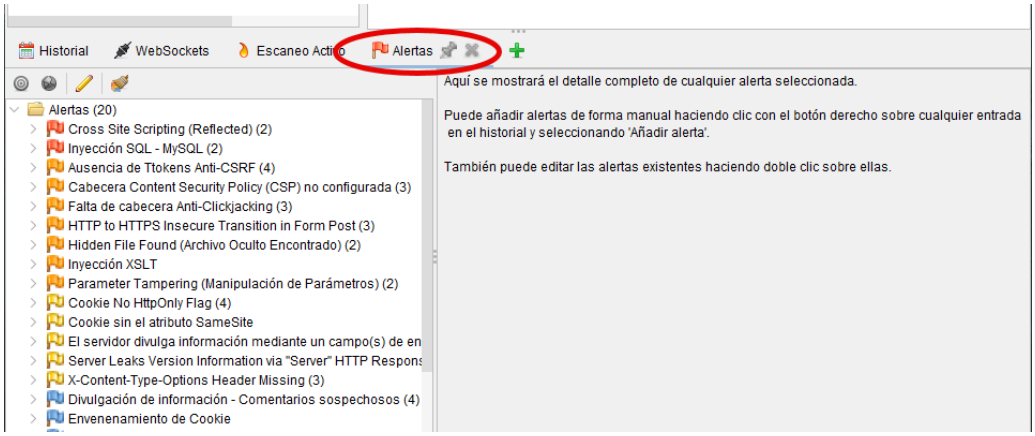


Figura 39: Pestaña de Alertas – Diseño del autor.

Al ingresar a un Alerta encontrado, se podrá observar la información de su vulnerabilidad; como el tipo de Alerta, la URL, el Riesgo, la Confianza, una Descripción, las posibles soluciones, entre otras. (Figura 40).

Figura 40: Descripción de un Alerta.

Entre los alertas, se pueden encontrar varios tipos que indican posibles vulnerabilidades. Entre los más comunes son:

1. **Injection (Inyección)**
    - **Descripción**: Incluye SQL Injection, Command Injection, LDAP Injection, etc. Estas vulnerabilidades permiten a un atacante inyectar comandos maliciosos en una aplicación.
    - **Tratamiento**:
        - Validar y sanitizar todas las entradas del usuario.
        - Utilizar consultas preparadas y declaraciones parametrizadas.
        - Implementar prácticas seguras de codificación.
2. **Cross-Site Scripting (XSS)**
    - **Descripción**: Permite a los atacantes inyectar scripts maliciosos en páginas web vistas por otros usuarios.
    - **Tratamiento**:
        - Escapar y codificar correctamente las entradas y salidas.
        - Utilizar bibliotecas de seguridad para prevenir XSS.
        - Implementar Content Security Policy (CSP).
3. **Broken Authentication and Session Management (Gestión de Sesiones y Autenticación Rota)**
    - **Descripción**: Vulnerabilidades en la autenticación y la gestión de sesiones pueden permitir a los atacantes robar contraseñas, claves de sesión, etc.
    - **Tratamiento**:
        - Implementar autenticación fuerte y segura.
        - Usar HTTPS para todas las conexiones.
        - Establecer tiempos de expiración para las sesiones.

4. **Insecure Direct Object References (Referencias Directas a Objetos Inseguras)**
    - **Descripción**: Permiten a un atacante acceder a objetos internos (archivos, directorios, registros de base de datos) mediante la manipulación de referencias.
    - **Tratamiento**:
        - Implementar controles de acceso adecuados.
        - Validar y verificar todas las referencias a objetos.
5. **Security Misconfiguration (Configuración de Seguridad Incorrecta)**
    - **Descripción**: Configuraciones incorrectas o inseguras pueden exponer la aplicación a ataques.
    - **Tratamiento**:
        - Revisar y ajustar las configuraciones de seguridad regularmente.
        - Eliminar configuraciones innecesarias y deshabilitar funcionalidades no utilizadas.
        - Mantener el software actualizado con los últimos parches de seguridad.
6. **Sensitive Data Exposure (Exposición de Datos Sensibles)**
    - **Descripción**: Datos sensibles (como información personal, números de tarjetas de crédito, etc.) expuestos sin la protección adecuada.
    - **Tratamiento**:
        - Encriptar datos sensibles en tránsito y en reposo.
        - Utilizar protocolos seguros (como TLS) para la transmisión de datos.
        - Implementar controles de acceso estrictos para datos sensibles.

7. **Cross-Site Request Forgery (CSRF)**
    - **Descripción**: Permite a un atacante realizar acciones no autorizadas en una aplicación en nombre de un usuario autenticado.
    - **Tratamiento**:
        - Implementar tokens anti-CSRF en formularios y solicitudes.
        - Validar las solicitudes del lado del servidor.
        - Utilizar técnicas de verificación del origen de la solicitud.
8. **Components with Known Vulnerabilities (Componentes con Vulnerabilidades Conocidas)**
    - **Descripción**: Uso de bibliotecas, frameworks o componentes con vulnerabilidades conocidas.
    - **Tratamiento**:
        - Mantener un inventario de todos los componentes utilizados.
        - Actualizar regularmente los componentes y aplicar parches de seguridad.
        - Utilizar herramientas de gestión de dependencias y monitoreo de vulnerabilidades.
9. **Unvalidated Redirects and Forwards (Redirecciones y Adelantos No Validados)**
    - **Descripción**: Permite a un atacante redirigir a los usuarios a sitios maliciosos o adelantar a recursos no autorizados.
    - **Tratamiento**:
        - Validar y verificar todas las redirecciones y adelantos.
        - Evitar el uso de entradas del usuario para determinar destinos de redirección.

Para tratar adecuadamente estas alertas, es fundamental tener un enfoque sistemático en la seguridad del desarrollo de software.

2.2.1.12. Generar informes

En ZAP puede generar informes detallados sobre las pruebas de seguridad realizadas en una aplicación web. Estos informes son útiles para documentar los hallazgos de seguridad, comunicarlo a los interesados y tomar acciones sobre ellos.

Para generar el informe, en la Barra de Menú, se debe hacer clic en "Informe" y en la lista seleccionar la opción "Generar informe" (Figura 41).

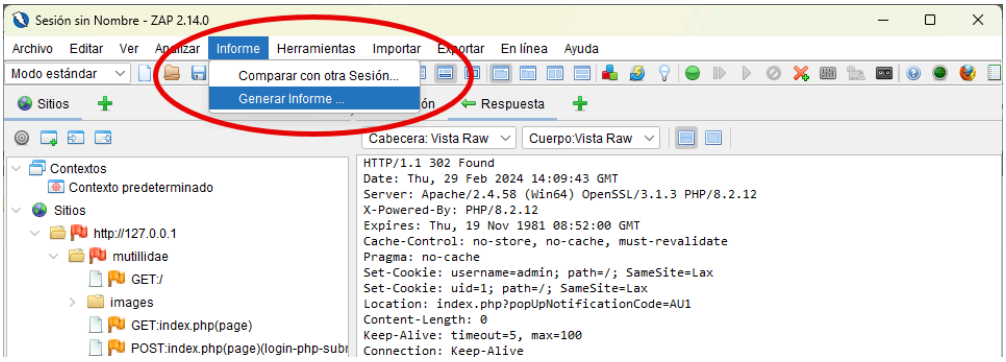


Figura 41: Opción de Generar informe en barra de menú – Diseño del autor.

Se abrirá una ventana donde se debe especificar los detalles del informe. Estará compuesto por diferentes pestañas para especificar la información que guardará dicho informe:

- **Ámbito**: En esta pestaña se coloca el Título del informe, el nombre, la ubicación, una descripción y el sitio (Figura 42).

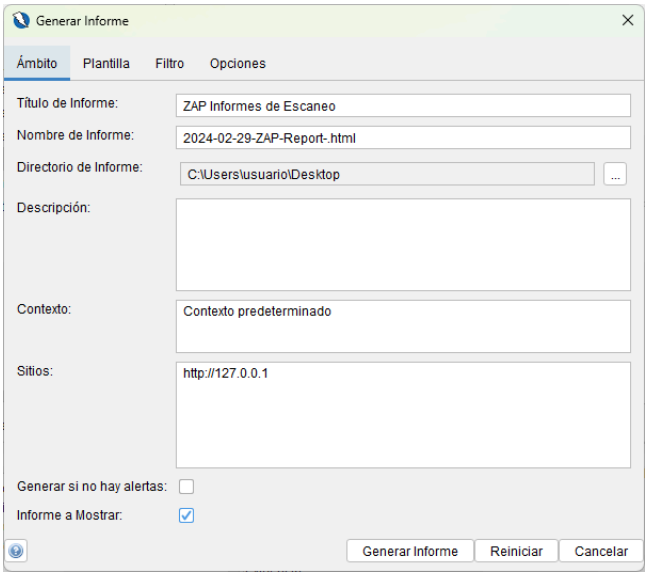


Figura 42: Pestaña de Ámbito en ventana Generar informe – Diseño del autor.

- **Plantilla**: Aquí se especifica el formato en que se generará el informe, se puede seleccionar el formato del informe (PDF, HTML, XML, etc.), el tema que tendrá y qué secciones interesa mostrar (Figura 43).

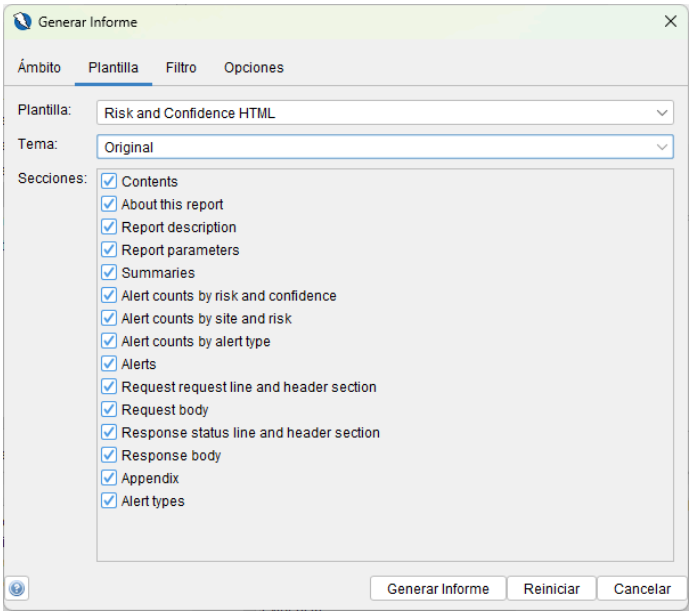


Figura 43: Pestaña Plantilla en ventana Generar informe – Diseño del autor.

- **Filtro**: Aquí se debe seleccionar, qué niveles de riesgo que contendrá el informe (Figura 44).

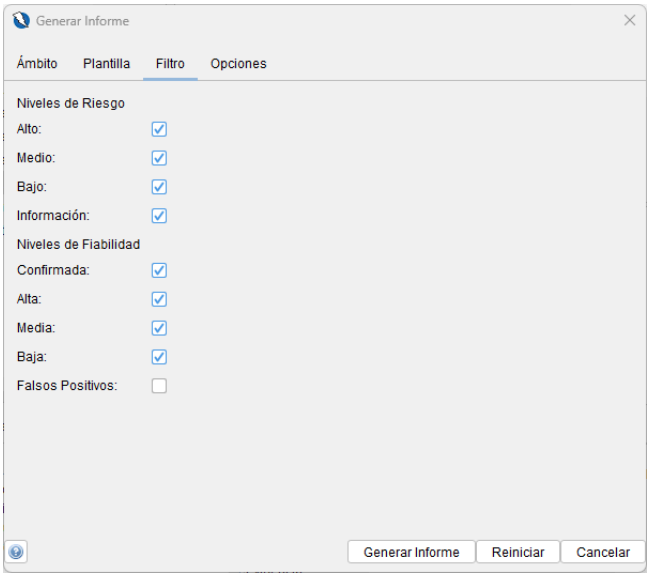


Figura 44: Pestaña Filtro en ventana Generar informe – Diseño del autor

- **Opciones**: En las opciones se especifica qué patrón tendrá la plantilla para el nombre que se va a generar. Además, la ubicación donde contendrá las plantillas (Figura 45).

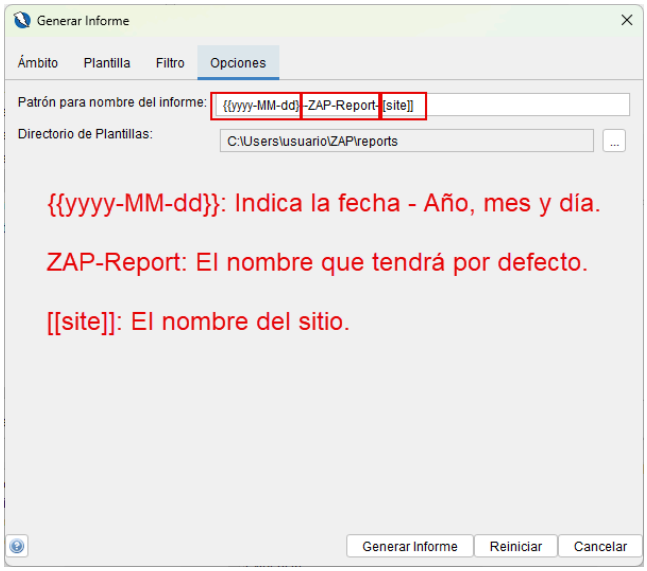


Figura 45: Pestaña Opciones en ventana Generar informe – Diseño del autor.

Luego de especificar toda la información, para finalizar solo se debe hacer clic en el botón “Generar informe”.

Cuando se genera un informe en OWASP ZAP, este incluirá detalles sobre los hallazgos de seguridad encontrados durante las pruebas de seguridad realizadas en una aplicación web. Los elementos típicos que pueden incluirse en un informe generado por ZAP son:

- **Contents (Contenidos)**: Muestra el índice con todos los contenidos que tendrá el informe.
- **About this report (Acerca del informe):** Contiene los parámetros de reporte, como el contexto, los sitios escaneados, y los niveles de riesgos y confidencialidad.
- **Summaries (Sumarios):** Muestra diferentes tablas con los niveles de riesgos y confidencialidad, de riesgos de cada sitio y los diferentes tipos de alertas.
- **Alerts (Alertas):** Ordena de forma ascendente de nivel alto al bajo los riesgos encontrados, mostrando la información específica sobre los hallazgos de seguridad encontrados, incluyendo detalles sobre la vulnerabilidad, su gravedad, ubicación en la aplicación y recomendaciones para su corrección.
- **Appendix (Apéndice):** Es una sección adicional que proporciona detalles complementarios o información de referencia que puede ser útil para comprender completamente los hallazgos de seguridad, las pruebas realizadas o cualquier otro aspecto relevante relacionado con la evaluación de seguridad de una aplicación web.

Los informes generados por OWASP ZAP están dirigidos a diferentes audiencias dentro de una organización, cada una con intereses y responsabilidades particulares. Las principales audiencias y cómo los informes pueden ser útiles para cada una son:

- **Desarrolladores**: Ayudan a los desarrolladores a entender las vulnerabilidades específicas en el código y cómo corregirlas.
- **Equipos de Seguridad (Security Analysts)**: Evalúan la seguridad general de la aplicación y asegurarse de que se implementen las mejores prácticas.
- **Gerentes de Proyecto**: Supervisan el progreso del proyecto y asegurarse de que las cuestiones de seguridad se están abordando adecuadamente.
- **Ejecutivos y Alta Dirección**: Aseguran que la organización cumple con las normativas de seguridad y mitigar riesgos potenciales.
- **Compliance Officers (Oficiales de Cumplimiento)**: Aseguran que la organización cumple con las regulaciones y estándares de seguridad aplicables.
- **Testers**: Verifican que las correcciones de seguridad se han implementado correctamente y que no se han introducido nuevas vulnerabilidades.

Los informes de OWASP ZAP se pueden personalizar según la audiencia y los objetivos específicos, proporcionando la información necesaria de manera clara y comprensible para cada grupo.

2.2.1.13. Contextos

Se refiere a un conjunto de configuraciones y opciones que definen el alcance de las pruebas de seguridad realizadas en una aplicación web específica. Los contextos permiten a los usuarios organizar y controlar las pruebas de seguridad de manera más efectiva al limitar el escaneo y las pruebas a un conjunto específico de URL, páginas web o funcionalidades de la aplicación.

Algunas características importantes de los contextos en OWASP ZAP incluyen:

1. **URL y Rutas Específicas**: Los contextos pueden incluir una lista de URL o rutas específicas que se deben incluir en las pruebas de seguridad. Esto permite a los usuarios enfocarse en partes específicas de la aplicación que son más críticas o relevantes desde el punto de vista de seguridad.
2. **Configuraciones de Autenticación y Sesión**: Los contextos pueden incluir configuraciones relacionadas con la autenticación y la gestión de sesiones, como credenciales de inicio de sesión, cookies de sesión, tokens de autenticación, entre otros.
3. **Exclusiones**: Los contextos también pueden definir URL, rutas o parámetros que deben excluirse de las pruebas de seguridad. Esto es útil para evitar escanear partes de la aplicación que no son relevantes para la evaluación de seguridad o que podrían generar falsos positivos.
4. **Configuraciones de Autenticación Personalizadas**: Los usuarios pueden configurar contextos con métodos de autenticación personalizados, como tokens de API, certificados SSL o sistemas de autenticación personalizados, para permitir el acceso a partes protegidas de la aplicación durante las pruebas.

Es decir que proporcionan una forma flexible y granular de definir el alcance de las pruebas de seguridad en una aplicación web, lo que permite a los usuarios personalizar y controlar con precisión qué partes de la aplicación se evalúan y cómo se realizan las pruebas. Esto es fundamental para realizar evaluaciones de seguridad efectivas y obtener resultados relevantes y significativos.

Para incluir el contexto en la aplicación ZAP, en la Barra de menú, hacer clic en “Archivo” y luego en “Propiedades de sesión” (Figura 46).

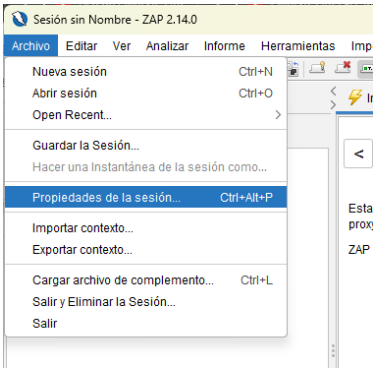


Figura 46: Propiedades de la sesión en Barra de menú – Diseño del autor.

Se abrirá una ventana con todas las propiedades de la aplicación. En la parte de Contextos, en el Contexto Predeterminado, se debe añadir el contexto que se desea incluir. Haciendo clic en el botón "Añadir". Se ingresa la URL y se hace clic en "Añadir", se coloca ".*" al final de la URL, para incluir todo el contenido del sitio (Figura 47).

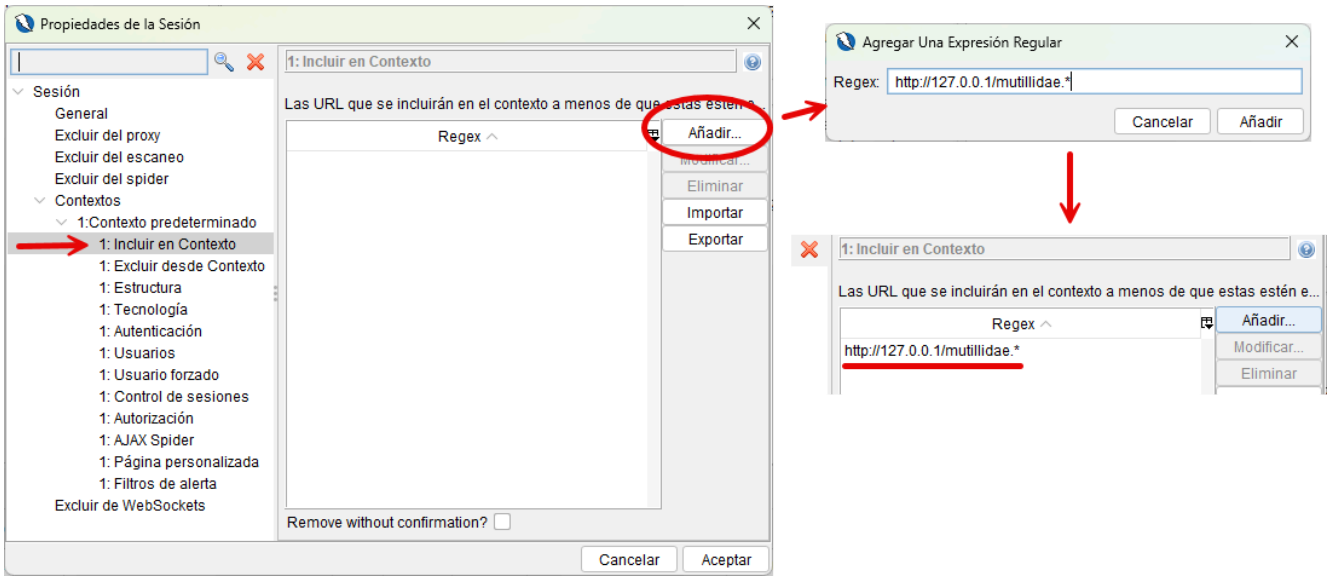


Figura 47: Añadir contexto a la sesión – Diseño del autor.

Otra de las maneras de poder agregar un contexto, en los sitios del Árbol, clic derecho en la URL del sitio que se desea agregar, seleccionar la opción “Incluir contexto” y luego en “Contexto predeterminado” (Figura 48).

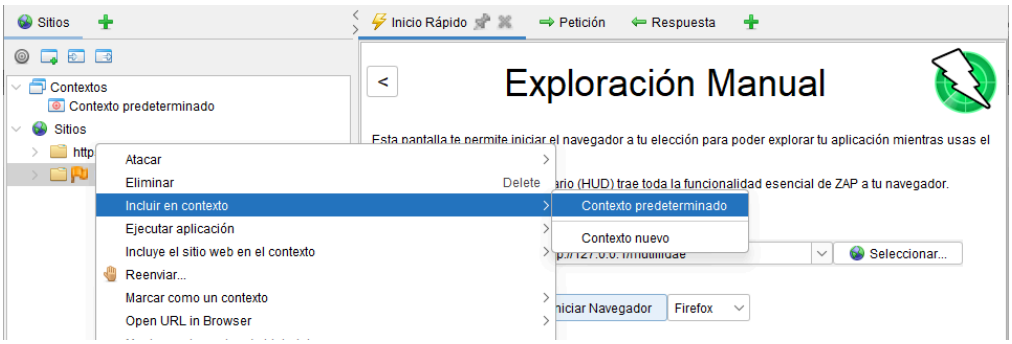


Figura 48: Añadir contexto desde el árbol – Diseño del autor.

Una vez agregado, se podrá observar un círculo rojo en la carpeta del icono de la URL del sitio en el Árbol, esto indicará que el contexto de agregó correctamente (Figura 49).

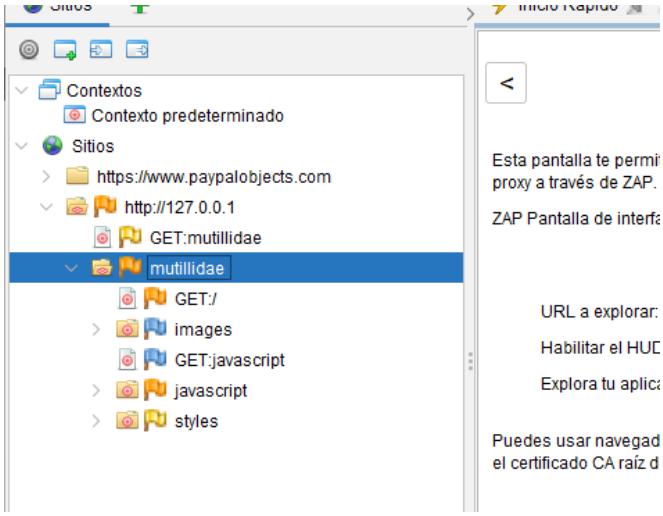


Figura 49: Ícono rojo en la URL del árbol – Diseño del autor.

2.2.1.14. Ámbito

El Ámbito es el conjunto de URL y dominios que se incluyen en las pruebas de seguridad realizadas por la herramienta. Definir el alcance correctamente es fundamental para asegurarse

de que ZAP evalúe únicamente las partes de la aplicación web que se pretenden analizar y evitar escanear accidentalmente sistemas externos o no relacionados.

Para incluir al ámbito, en la Barra de Menú, en Archivo se ingresa a las Propiedades de la sesión. En la propiedad "Contexto", en "Contexto predeterminado" se verifica que la casilla de Ámbito debe estar marcada, lo que indica que el contexto incluido estará dentro del ámbito de las pruebas que se realizarán (Figura 50).

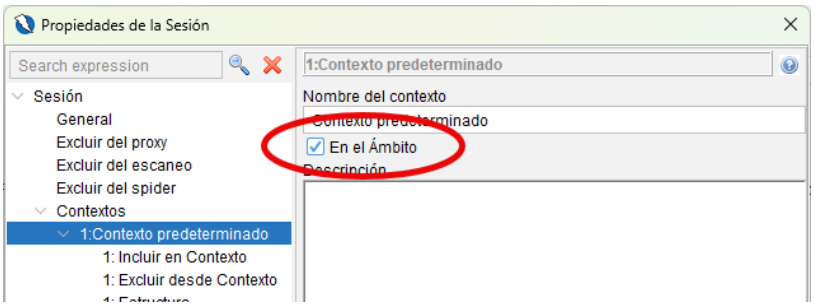


Figura 50: Casilla de inclusión del Ámbito en el contexto – Diseño del autor.

En la parte superior del Árbol, se puede observar un botón circular de color negro que, al hacer clic en él, se podrá ver solo las URL dentro del Ámbito, al activarlo cambia a color rojo (Figura 51).

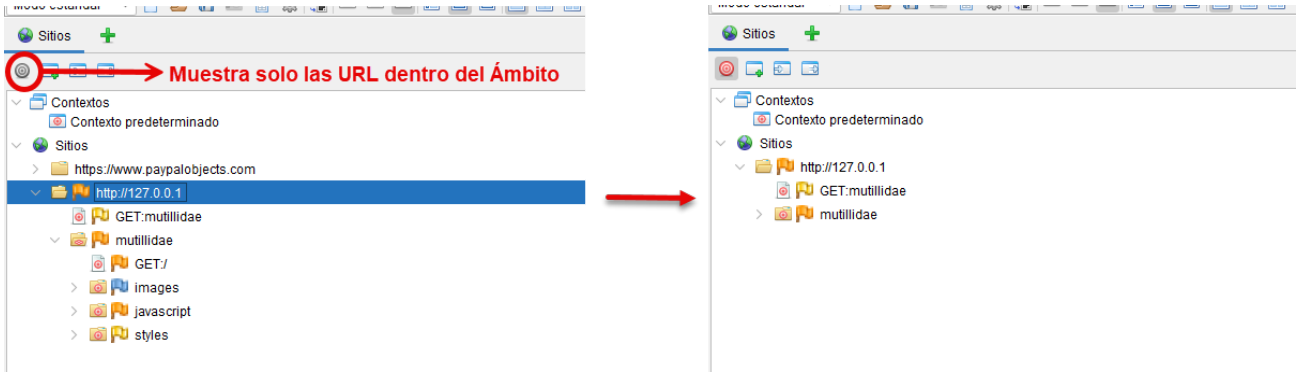


Figura 51: Cambio de estado de icono para visualizar el ámbito – Diseño del autor.

Otra manera de activarlo se podrá realizar desde el Historial en la ventana de información (Figura 52).

Figura 52: Activación del Ámbito desde la pestaña Historial – Diseño del autor.

Configurar el Ámbito adecuadamente es esencial para asegurar que las pruebas de seguridad sean efectivas y relevantes. Un alcance mal definido puede llevar a pruebas incompletas, resultados inexactos o impactos no deseados en sistemas externos. Por lo tanto, es importante revisar y ajustar el Ámbito según sea necesario para cada evaluación de seguridad.

2.2.1.15. Modos

ZAP permite realizar diferentes acciones dependiendo del "modo" seleccionado. Para ello, en la Barra de Herramientas se selecciona el modo deseado (Figura 53).

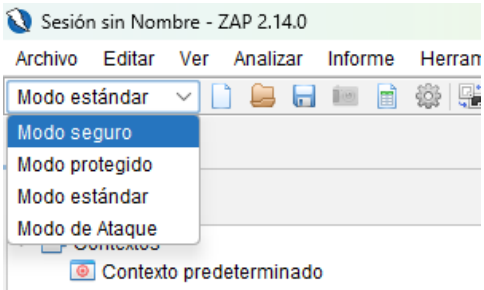


Figura 53: Modos de escaneo en ZAP – Diseño del autor.

Estos modos son:

- **Modo Seguro:** Al seleccionar este modo no habrá ninguna operación potencialmente peligrosa que pueda realizarse. Por lo que no se podrán realizar ataques ya que estarán deshabilitados.
- **Modo Protegido:** Con este modo se podrán realizar acciones potencialmente peligrosas en las URL, pero solo si se encuentran dentro del ámbito. Por lo que se podrán todos los ataques disponibles.
- **Modo Estándar:** Se podrán realizar todas las acciones sin limitaciones.
- **Modo de Ataque**: Este modo proporciona herramientas para llevar a cabo diferentes tipos de ataques contra la aplicación objetivo con el fin de evaluar su resistencia a diversas amenazas.

2.2.1.16. Editor Manual de Peticiones

El Editor de peticiones es una función que permite modificar manualmente las solicitudes HTTP que ZAP intercepta mientras se navega por una aplicación web. Esto brinda un control completo sobre las solicitudes que se envían al servidor, lo que es útil para probar la seguridad de la aplicación y encontrar posibles vulnerabilidades.

Para abrir el Editor, se dirige a la URL donde se desea realizar las acciones (en el Historial o en el Árbol), se debe hacer clic secundario y seleccionar “Reenviar…” en la lista (Figura 54) y se abrirá la ventana del Editor (Figura 55).

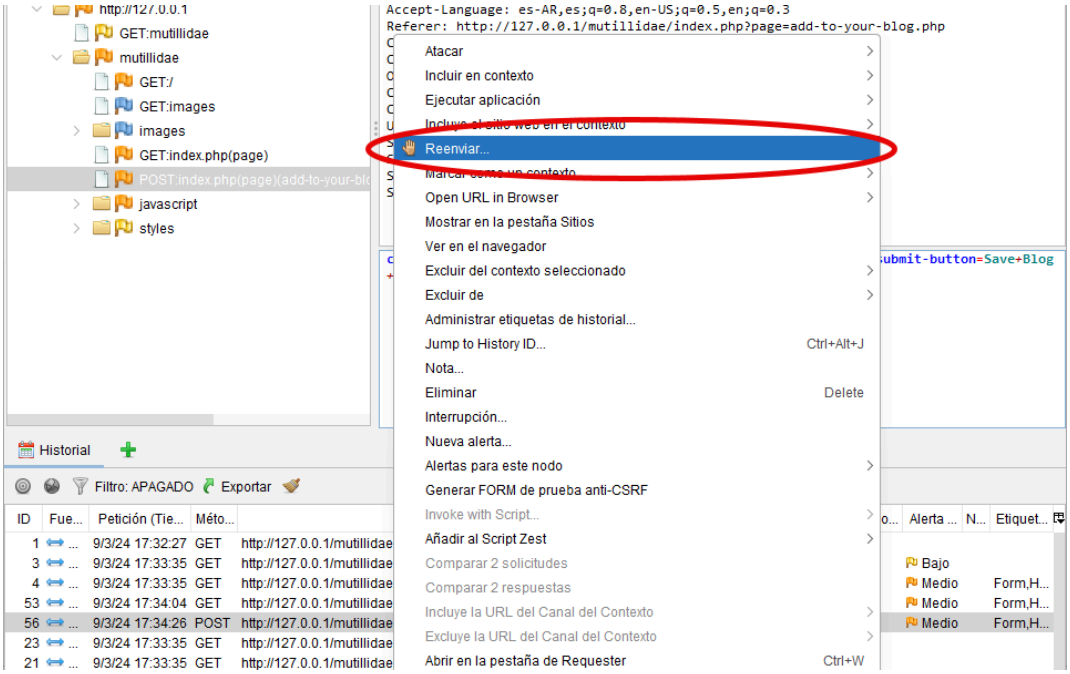


Figura 54: Abrir Editor desde Árbol o Historial – Diseño del autor.

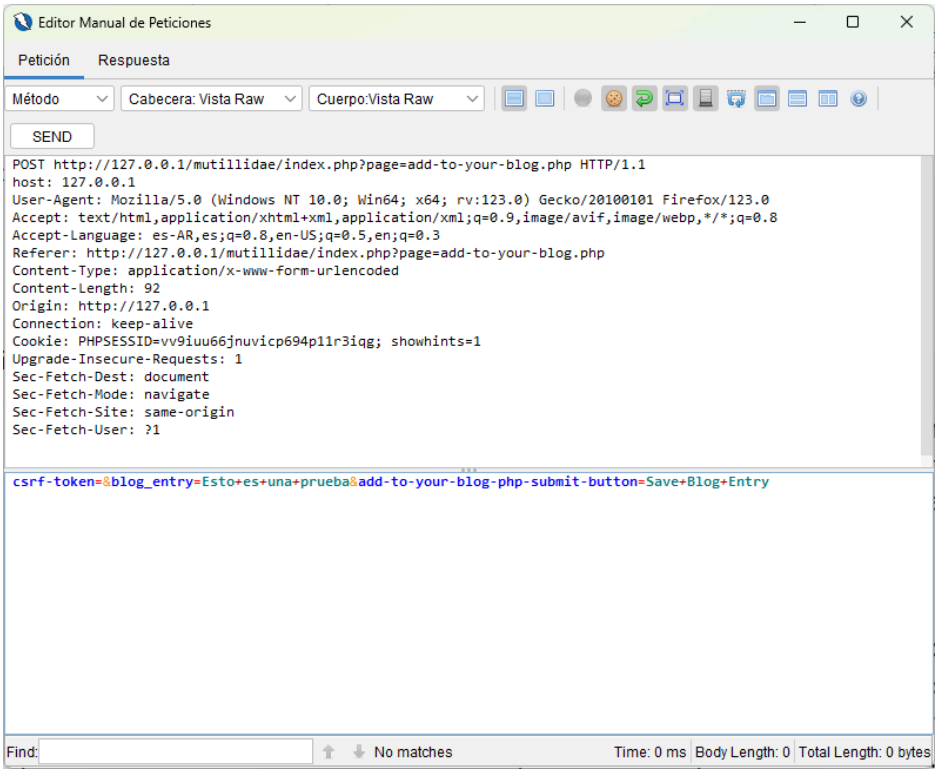


Figura 55: Ventana de Editor Manual de Peticiones – Diseño del autor.

El Editor manual de peticiones permite realizar las siguientes acciones:

- **Modificar Parámetros:** Permite editar los parámetros de las solicitudes, como campos de formulario, parámetros de URL, cookies, cabeceras HTTP, entre otros. Esto te permite probar cómo la aplicación responde a diferentes valores y datos de entrada.
- **Agregar o Eliminar Parámetros:** Permite agregar nuevos parámetros a las solicitudes o eliminar los existentes según sea necesario. Esto te permite probar escenarios específicos o eliminar datos que no son relevantes para tus pruebas.
- **Repetir Solicitudes:** Permite reenviar las solicitudes modificadas al servidor para ver cómo responde la aplicación a los cambios realizados. Esto te permite probar posibles vulnerabilidades o errores de seguridad al manipular las solicitudes.
- **Guardar y Cargar Solicitudes:** Permite guardar las solicitudes modificadas en archivos para referencia futura o cargar solicitudes guardadas anteriormente para realizar pruebas repetidas.

El Editor de Solicitudes es una herramienta poderosa que brinda flexibilidad y control sobre las pruebas de seguridad que realizas en una aplicación web. Permite realizar pruebas detalladas y específicas, así como también probar diferentes escenarios y situaciones para identificar posibles vulnerabilidades y riesgos de seguridad. Es importante utilizar esta función con responsabilidad y ética, asegurándo de obtener el consentimiento del propietario de la aplicación antes de realizar cualquier prueba de seguridad.

2.2.1.17. Reglas de Escaneo Pasivo

El escaneo pasivo es el escaneo el cual ZAP realiza automáticamente en segundo plano. Es una técnica utilizada para identificar vulnerabilidades y riesgos de seguridad en una aplicación web

al observar y analizar el tráfico HTTP que pasa a través del proxy de ZAP. Durante este tipo de escaneo, ZAP monitorea y registra todas las solicitudes y respuestas HTTP que ocurren entre el cliente (navegador web) y el servidor web, sin interactuar directamente con la aplicación.

Este escaneo posee ciertas reglas que definen que tipo de vulnerabilidades serán verificadas al escanear pasivamente una aplicación web.

- **Errores de aplicación**: Verifica las respuestas del servidor, como el error HTTP 500 (errores internos del servidor)
- **Cookies sin un atributo SameSite**: (Cuando se utilizan sitios de terceros para robar información a través de las cookies). Informa cualquier cookie que no tenga los mismos atributos del sitio configurado para que no realicen ataques de falsificación de solicitudes.
- **Cookies HttpOnly**: Indica si las cookies configuradas están siendo accedidas por el lado del cliente.
- **Definición de Cross Domain Script Include**: Verifica si la aplicación web permite la inclusión de scripts maliciosos desde un dominio externo.
- **Información divulgada en la URL**: Verifica la existencia de información divulgada en una URL puede incluir datos personales identificables, como nombres, direcciones y números de teléfono. También puede incluir información confidencial, como números de tarjetas de crédito y contraseñas.
- **Información divulgada en Comentarios**: Identifica los comentarios que contengan detalles potencialmente confidenciales. Algunos comentarios que poseen valores de identificación, como usuario y contraseña.

Dentro de la aplicación en la Barra de Menú, Herramientas, Opciones y en la lista, se debe buscar la opción “Reglas de la exploración pasiva” (Figura 56).

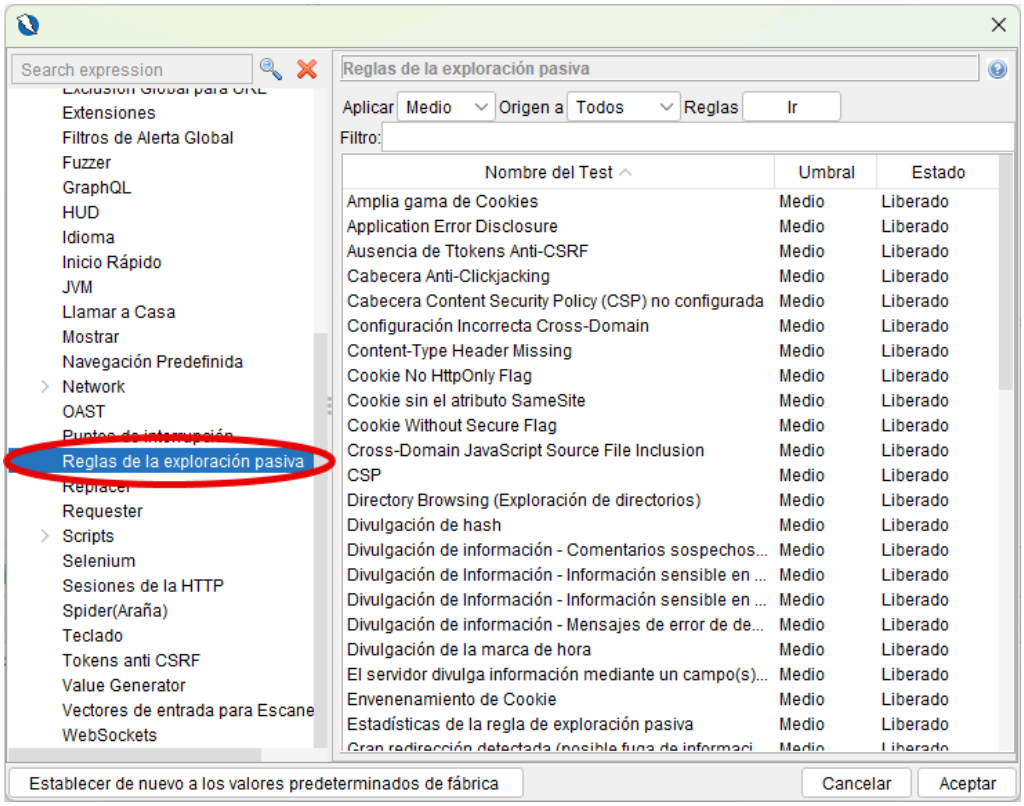


Figura 56: Reglas de la exploración pasiva en la ventana de opciones – Diseño del autor.

Allí se encontrarán todas las reglas que se configuraron para el escaneo pasivo. En la tabla estará el "Umbral" (se refiere al nivel de severidad que se establece como criterio para determinar si una vulnerabilidad detectada durante el escaneo pasivo debe ser reportada o no) de cada regla, por lo que se podrá cambiar su nivel dependiendo de las necesidades del escaneo, además de cambiar su valor a: Apagado, Defecto, Medio y Alto. Por el nivel del Umbral tendrá la priorización de la búsqueda. En la columna "Estado" se podrá saber si las reglas están Liberadas, en su versión Beta o Desconectadas. También posee un filtro para la búsqueda por medio del nombre de la regla, el umbral, entre otros.

Al desplazarse hasta la opción "Escáner pasivo" en las opciones de la aplicación, se podrán realizar diferentes configuraciones para el Escaneo pasivo, como activar la opción de que solo

analice los mensajes dentro del Ámbito, incluir el tráfico del Fuzzer, número de semillas, cantidad de Alertas que pueda generar y el tamaño máximo del cuerpo de bytes para escanear (Figura 57)

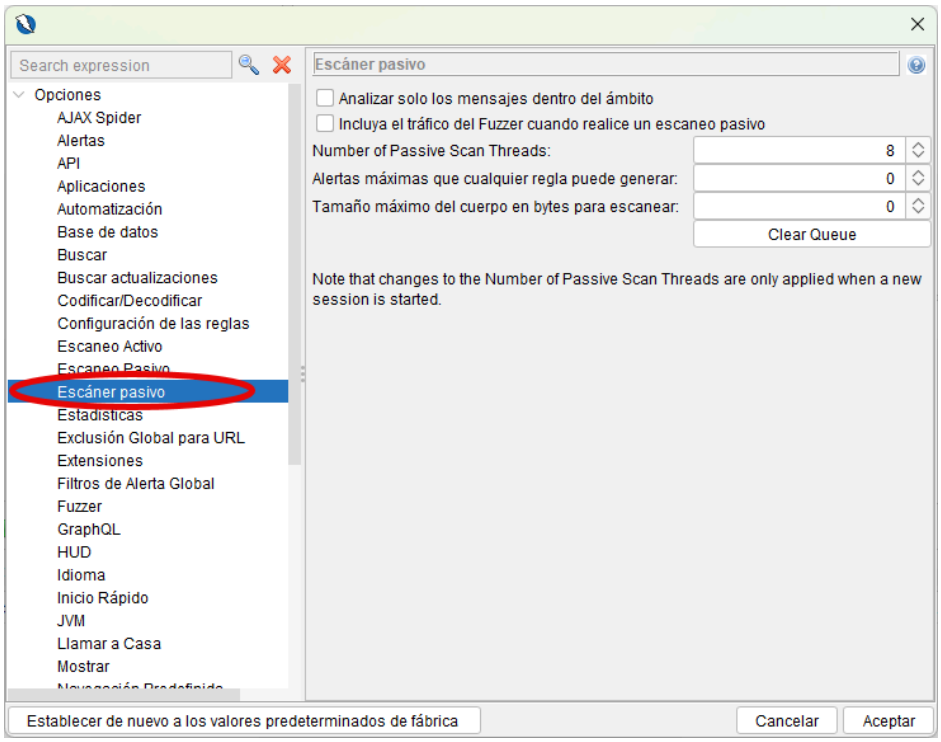


Figura 57: Escaneo pasivo en las opciones de la aplicación – Diseño del autor.

También se podrán observar las etiquetas del escaneo, para ello en las opciones de "Escaneo Pasivo". Las etiquetas son un breve fragmento de texto que se puede asociar con la solicitud. Se pueden agregar, modificar y eliminar etiquetas de alertas de posibles problemas, esto permite que estas etiquetas coincidan con expresiones regulares (Figura 58).

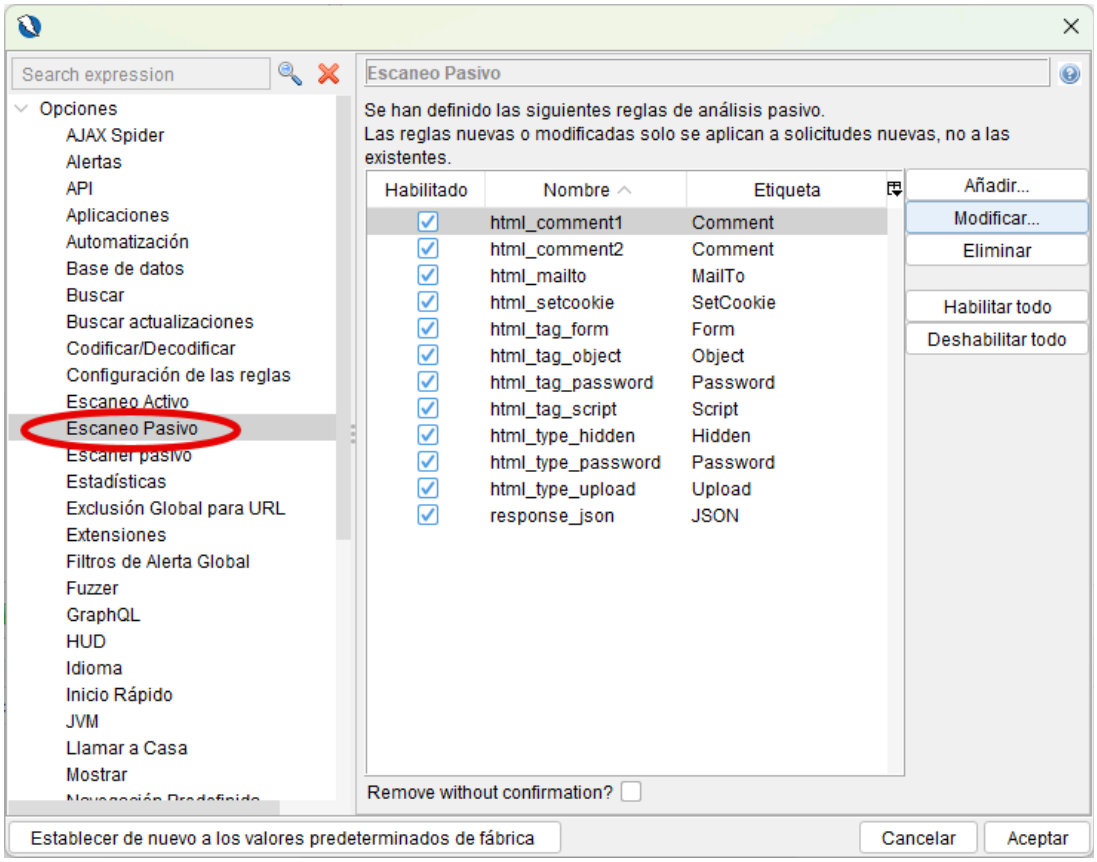


Figura 58: Reglas del escaneo pasivo – Diseño del autor.

2.2.1.18. Escaneo Activo

En el Escaneo Pasivo solo analiza las solicitudes y las respuestas y descubre cualquier vulnerabilidad. En cambio, en el Escaneo Activo:

- Ataca la aplicación bajo prueba.
- Encuentra vulnerabilidades potenciales usando conocidos ataques contra los objetivos seleccionados. Por lo que sólo podrá encontrar ciertos tipos de vulnerabilidades.
- Las vulnerabilidades lógicas como el control de acceso dañados, no serán encontrados por ningún escaneo activo o automatizado.
- Siempre se deben realizar pruebas de penetraciones manuales para el escaneo activo.

Reglas del Escaneo Activo definen qué tipos de vulnerabilidades serán verificadas cuando se realice un ataque en una aplicación. Algunos ejemplos de las reglas son:

- Fuga de información .htaccess: Verifica si hay archivos de acceso .htaccess desde la web que pueda filtrar información confidencial, como nombres de usuario o listado de directorios.
- Inyección de código: Es un tipo de ataque que permite a un atacante inyectar código que luego es interpretado/ejecutado por la aplicación. Este tipo de ataque se puede utilizar para obtener acceso a información restringida y aumentar privilegios.
- Inyección de comandos: Envía comandos del sistema operativo como parámetros de URL para determinar si la aplicación web está pasando o no entradas de usuarios no verificadas directamente al sistema operativo subyacente.
- Cross-Site Scripting (XSS): Es un tipo de vulnerabilidad de seguridad que permite a los atacantes inyectar código malicioso en un sitio web.
- Navegación de directorios: Es una falla de seguridad que permite a los atacantes ver el contenido de los directorios de un servidor web, incluso si no tienen permiso para hacerlo. Esto puede darles acceso a información confidencial, como archivos de configuración, bases de datos y código fuente.
- Inyecciones SQL: Busca comandos SQL maliciosos en la aplicación que puedan afectar la base de datos.

Para configurar el Escaneo Activo en la aplicación de ZAP, solo se debe dirigir a la opción “Escaneo Activo” de las Opciones en Herramientas en la Barra de Menú (Figura 59).

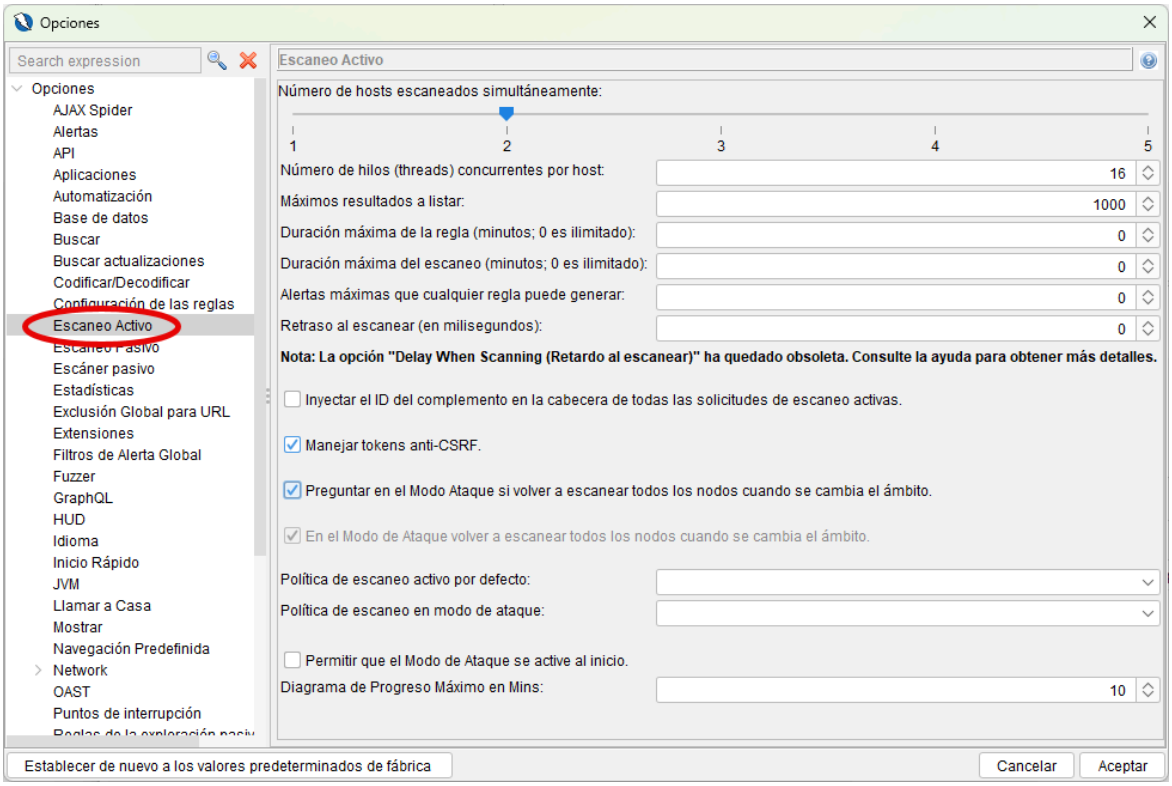


Figura 59: Opciones de Escaneo Activo – Diseño del autor.

Allí se podrá configurar: la cantidad de hosts escaneados simultáneamente, la cantidad máxima de hosts que se escanearán al mismo tiempo (aumentar esta opción puede perjudicar el rendimiento de la computadora), número máximo de hilos concurrentes por hots, duración máxima de la regla y escaneo, alertas máximas que puede generar y el retraso que puede generar en milisegundos. Además, se podrá activar la opción de inyectar el ID del comentario, manejar tokens anti-CSRF, preguntar en el Modo de Ataque si vuelve a escanear todos los nodos cuando se cambia el ámbito, las políticas, entre otros.

Pero para atacar, se deben configurar los Vectores, al desplazarse en la lista de opciones hasta llegar a “Vectores de entrada para escaneo activo”. Los vectores son los elementos específicos que la herramienta utiliza para realizar pruebas agresivas en una aplicación web y detectar posibles vulnerabilidades de seguridad. Estos vectores representan diferentes tipos de ataques

que ZAP puede llevar a cabo durante un escaneo activo para identificar vulnerabilidades comunes. Algunos ejemplos de vectores pueden ser: Objetivos inyectables (como datos POST, cabeceras HTTP, datos de las cookies, entre otros) y controladores de entradas (como las etiquetas XML, JSON, Google web toolkit, entre otros). Además, se podrá configurar qué parámetros debe ignorar el escáner, como por ejemplo, los ID de sesión de .NET o PHP (Figura 60).

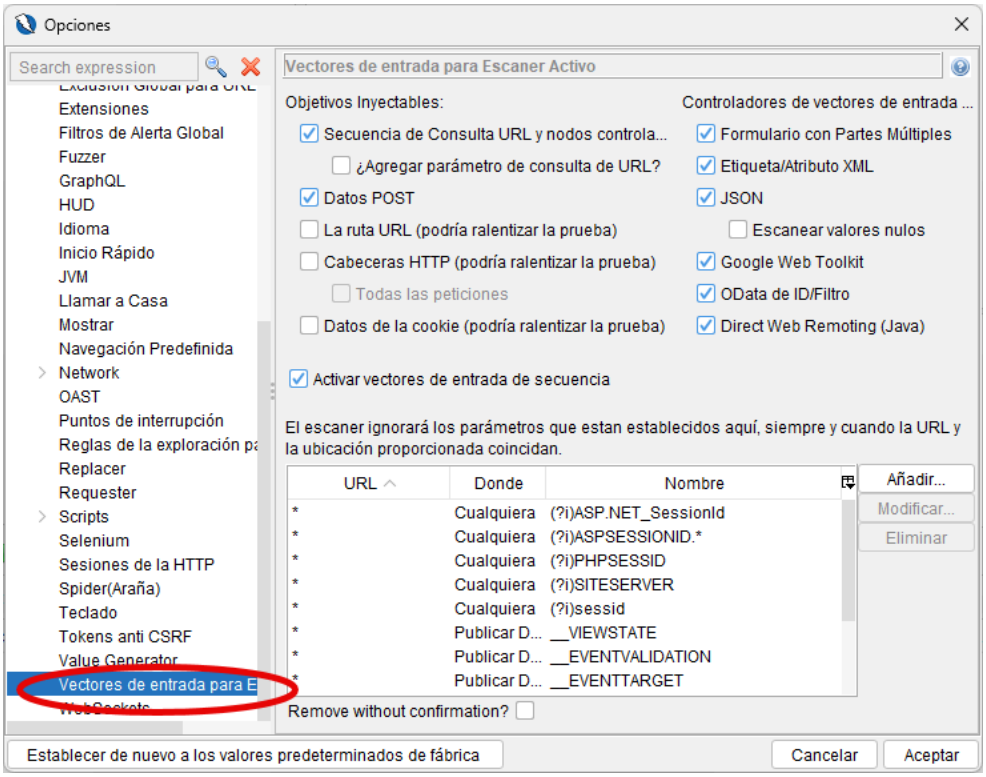


Figura 60: Vectores de entrada para escaneo activo – Diseño del autor.

2.2.1.19. Políticas de Escaneo Activo

Las políticas de escaneo activo son conjuntos de reglas predefinidas que dictan cómo se llevan a cabo las pruebas de seguridad durante un escaneo activo en una aplicación web. Estas políticas permiten a los usuarios personalizar y ajustar el comportamiento de ZAP durante el escaneo para

adaptarse a las necesidades específicas de la aplicación y las preferencias de seguridad del usuario.

Algunas características y funcionalidades de las políticas de escaneo activo en OWASP ZAP incluyen:

- **Selección de Reglas:** Las políticas de escaneo activo permiten a los usuarios seleccionar qué reglas específicas se aplicarán durante el escaneo para identificar vulnerabilidades de seguridad en la aplicación web. Estas reglas pueden incluir detección de inyecciones SQL, cross-site scripting (XSS), inseguridad de cookies, entre otros.
- **Niveles de Severidad**: Las políticas de escaneo activo permiten a los usuarios configurar el nivel de severidad deseado para las reglas aplicadas durante el escaneo. Esto permite enfocar el escaneo en áreas específicas de preocupación o riesgo dentro de la aplicación.
- **Personalización de Parámetros:** Las políticas de escaneo activo permiten a los usuarios personalizar los parámetros específicos de las reglas aplicadas durante el escaneo, como la profundidad de exploración, la frecuencia de las solicitudes, entre otros.
- **Creación de Políticas Personalizadas:** Además de las políticas predefinidas, ZAP permite a los usuarios crear políticas de escaneo activo personalizadas, donde pueden definir sus propias reglas y configuraciones específicas de acuerdo a las necesidades de seguridad de la aplicación.
- **Uso de Plantillas:** ZAP proporciona plantillas predefinidas para diferentes tipos de aplicaciones y tecnologías, que incluyen conjuntos específicos de reglas y configuraciones recomendadas para escaneos activos.

Para acceder a las políticas en la aplicación, en la Barra de Menú, en "Analizar", haciendo clic en "Reglas de Escaneo…". También se podrá ingresar desde la barra de herramientas y haciendo clic en el siguiente icono (Figura 61).

Figura 61: Opciones para acceder a las políticas – Diseño del autor.

Se abrirá una ventana del Gestor de Políticas de Escaneo. Inicialmente se verá la Política por defecto (Figura 62).

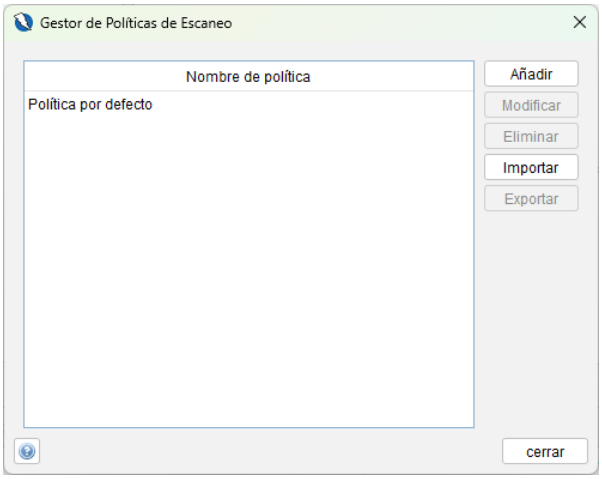


Figura 62: Gestor de Políticas de Escaneo – Diseño del autor.

Además, se podrá Modificar o Agregar una nueva política. Esta política tendrá todas las opciones activadas por defecto y todos los parámetros como: inyección, misceláneas[6], navegador del cliente, recopilación de información y seguridad del servidor, podrán ser modificados (Figura 63).

[6] Miscelánea: Puede referirse a un conjunto de funciones, variables o elementos de código que no pertenecen a una categoría específica o que no tienen una relación directa con el resto del código. Estas misceláneas pueden incluir funciones de utilidad, constantes, clases auxiliares, entre otros.

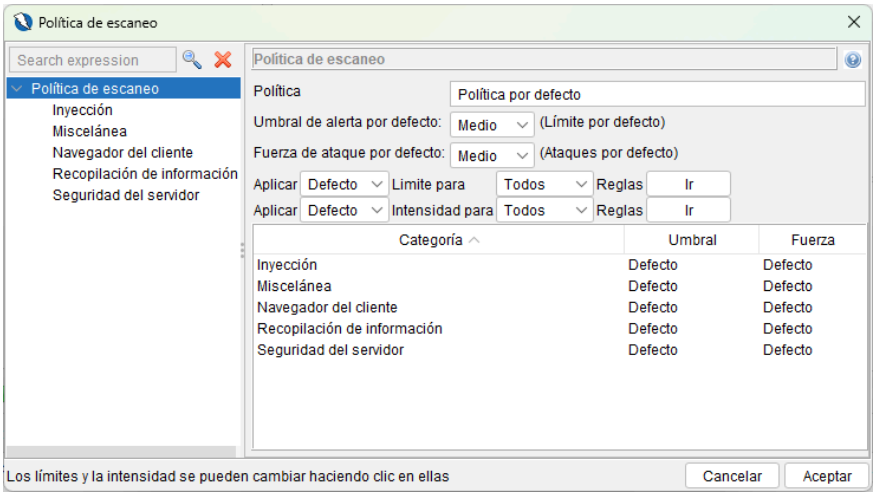


Figura 63: Políticas de escaneo – Diseño del autor.

2.2.1.20. Modo Ataque

El modo ataque es similar al escaneo activo, pero funciona de una manera diferente. Al dirigirse a la aplicación, en la barra de herramientas se podrá seleccionar el Modo de Ataque en la lista (Figura 64).

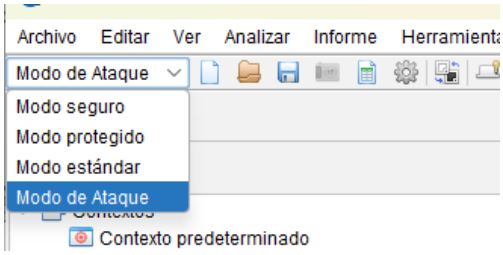


Figura 64: Modo de Ataque en la barra de herramientas – Diseño del autor.

Al hacer clic, se podrá observar un mensaje que informa si se desea escanear todos los nodos al cambiar de ámbito, por lo que es conveniente aceptar el mensaje (Figura 65).

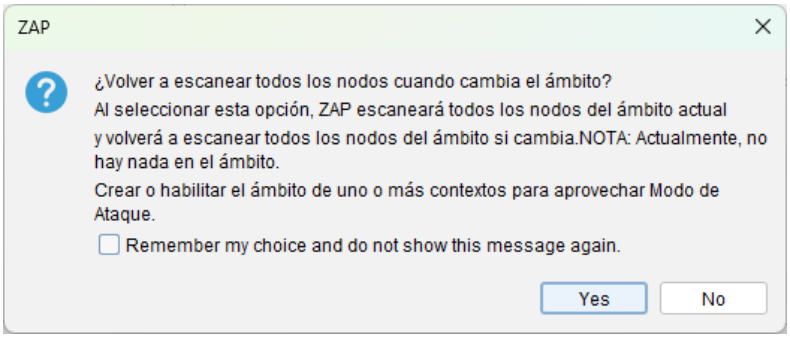


Figura 65: Mensaje si se desea escanear todos los nodos del ámbito – Diseño del autor.

Si se acepta el mensaje se observa que, en la ventana de información, se activa una pestaña llamada "Escaneo Activo" (Figura 66).

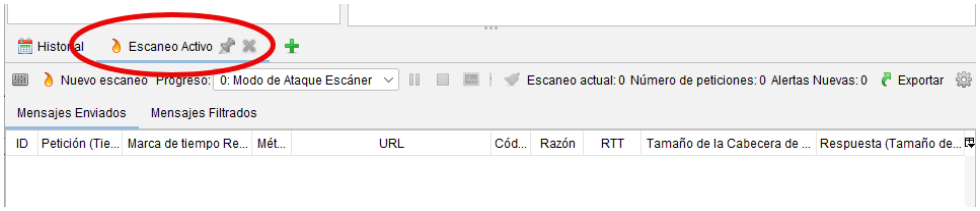


Figura 66: Pestaña de Escaneo Activo – Diseño del autor.

Al realizar un escaneo manual de un sitio se podrá observar que en la pestaña del Historial comienza a cargar información. Lo que resulta importante es cargar el ámbito y el contexto en las opciones para que el modo de ataque pueda funcionar.

En la pestaña de Escaneo Activo, que se inicia cuando se inicia el ataque, comienzan a cargar las solicitudes mientras se navega por sitio se agregó anteriormente en el contexto. Este modo es muy útil para centrarse en un área particular del sitio, cuando la aplicación o el sitio es realmente grande y por lo que en lugar de rastrear todos los nodos o URL rastrea solo el área específica (Figura 67).

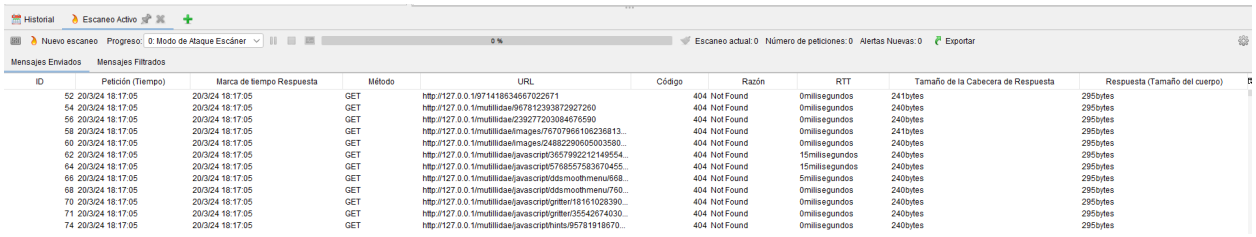


Figura 67: URLS específicas cargadas – Diseño del autor.

Al iniciar el escaneo se podrá observar una barra de progreso, por lo que nunca cargará debido a que el modo de ataque nunca terminará, finalizará cuando se deja de navegar por el sitio.

Se podrá tener diferentes políticas para el modo de ataque, esto es útil cuando se centra en algún tipo de vulnerabilidad específico en un área en particular del sitio.

Para poder agregar alguna política, se dirige a las opciones de la Aplicación, en Herramientas de la barra de herramientas. En la opción “Escaneo Activo”, en la parte de “Política de escaneo en modo de ataque”, y se selecciona de la lista la política que se desea incluir (Figura 68).

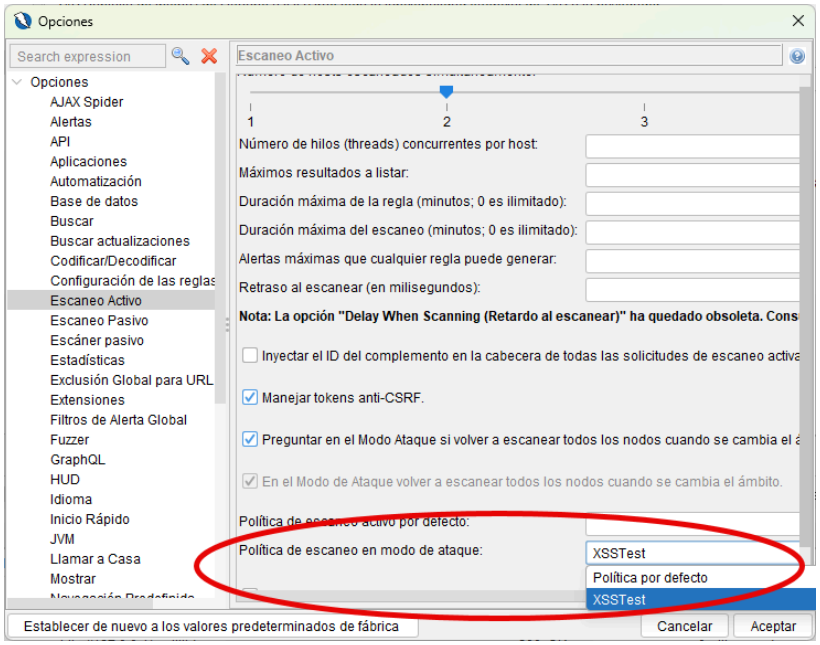


Figura 68: Opciones de Escaneo Activo – Diseño del autor.

Una vez cargadas las políticas, en el árbol donde se encuentra el sitio, haciendo clic con el secundario, se selecciona la opción “Incluir en contexto” el contexto cargado (Figura 69).

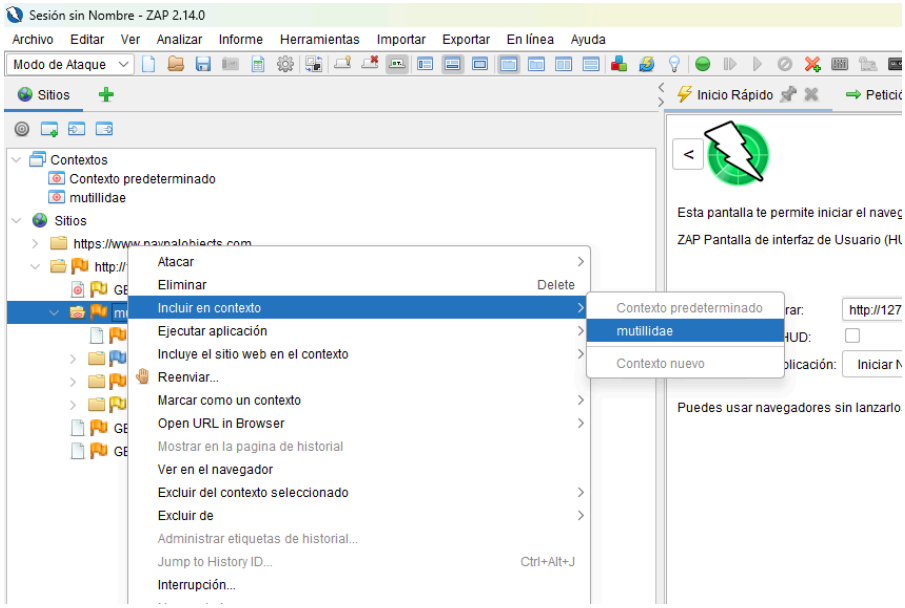


Figura 69: Incluir contexto a escaneo – Diseño del autor.

A continuación, comenzará a cargar en la pestaña de “Escaneo Activo” del Historial, todas las solicitudes por lo que sólo cargará las políticas seleccionadas anteriormente (Figura 70).

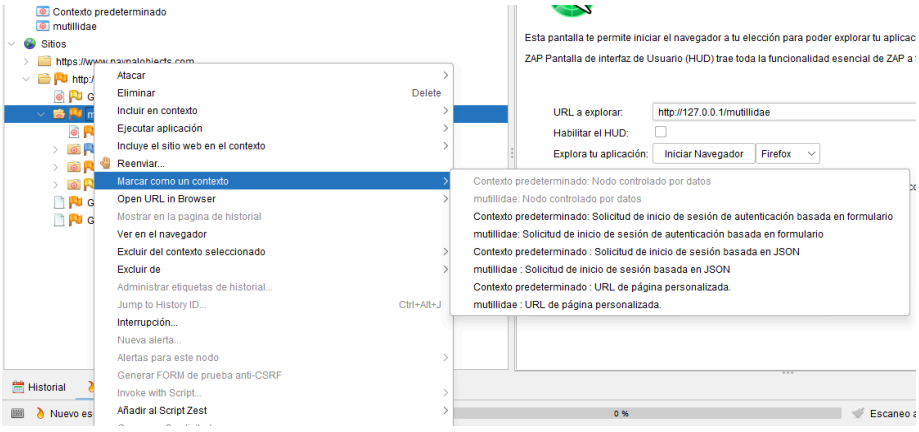


Figura 70: Marcar un contexto al escaneo – Diseño del autor.

Al navegar por el sitio, se podrá observar, los Alertas encontrados con las políticas que se han seleccionado anteriormente. Por lo que se podrá centrar en vulnerabilidades o áreas específicas del sitio para luego reportarlas (Figura 71).

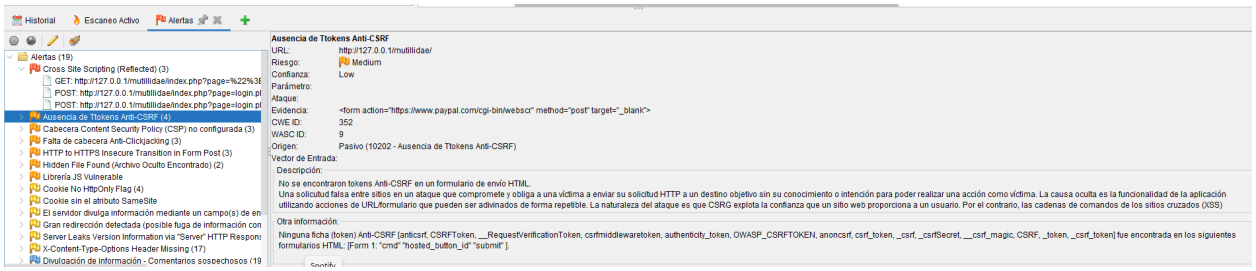


Figura 71: Alertas de vulnerabilidades o áreas específicas – Diseño del autor.

2.2.2. Análisis estático con OWASP Dependency-Check

OWASP Dependency check (Verificación de dependencia) es una herramienta de análisis de composición de software (SCA), (aunque puede considerarse parte del análisis estático ya que examina el código fuente, archivos de configuración y dependencias) intentado detectar vulnerabilidades divulgadas públicamente contenidas en las dependencias de un proyecto. Para ello, al aprovechar los identificadores de enumeración de plataforma común (CPE)[7], OWASP Dependency Check examina cada dependencia para determinar si es susceptible a alguna vulnerabilidad conocida. Lo cual genera un informe detallado que se puede utilizar como referencia y análisis adicional.

El OWASP Top 10 2021 contiene una entrada: A06:2021-Componentes vulnerables y obsoletos. Actualmente [12], Dependency Check se puede utilizar para escanear aplicaciones (y sus bibliotecas dependientes) para identificar cualquier componente vulnerable conocido.

[7] CPE: (Common Platform Enumeration - Enumeración de Plataforma Común) Es un sistema de identificación estándar utilizado en la industria de la tecnología de la información para identificar de manera única y unificada los productos y servicios de hardware y software.

OWASP como comunidad de desarrollo, incluye bibliotecas de terceros en la aplicación, que contiene vulnerabilidades publicadas bien conocidas (como las de la Base de datos nacional de vulnerabilidades [19]).

El motor central contiene una serie de analizadores que inspeccionan las dependencias del proyecto y recopilan información sobre las dependencias (lo que se conoce como evidencia dentro de la herramienta). Luego, la evidencia se utiliza para identificar la enumeración de plataforma común (CPE) para la dependencia dada. Si se identifica un CPE, se incluye en un informe una lista de entradas de exposición y vulnerabilidad.

Sus principales características son:

- Como la Fundación OWASP es una organización sin fines de lucro, la herramienta es gratuita. Los desarrolladores pueden descargar la herramienta y comenzar a usarla como parte de su paquete de seguridad.
- Es fácil de comenzar y no requiere revisión de documentación, capacitación o certificación.
- Ofrece múltiples opciones de informes para que los desarrolladores verifiquen y rectifiquen las vulnerabilidades de código abierto. Las funciones de exportación de la herramienta permiten a los equipos crear informes de vulnerabilidad que se centran en sus métricas clave.
- Es compatible con muchos lenguajes, tecnologías y plataformas populares, incluidos:
    - Soporte completo para productos basados en Java y .NET
    - Soporte experimental para proyectos Node.js, Ruby y Python
    - Parcialmente compatible con C y C++

- Se integra con Maven, Jenkins y Gradle mediante complementos y puede ejecutarse a través de la CLI como una tarea Ant.

2.2.2.1. Cómo Dependency-Check detecta vulnerabilidades

WASP Dependency-Check identifica vulnerabilidades mediante analizadores. Estos son proyectos dedicados de código abierto que ejecutan todo el proceso de escaneo de dependencias. Esencialmente una herramienta de rastreo, los analizadores son responsables de examinar cada paquete de datos para verificar su relevancia y registrar su información. Se pueden utilizar para la recuperación de datos y el escaneo de errores.

2.2.2.2. Cómo funcionan los analizadores

Dependency-Check escanea archivos y recopila información sobre las dependencias del proyecto a través de una serie de analizadores. Esta información recopilada, conocida como evidencia, se divide en tres categorías: información de proveedor, plataforma y versión. Luego se etiqueta con un nivel de confianza (bajo, medio, alto y más alto). La calificación de confianza es un método para señalar la vulnerabilidad potencial que debe verificarse.

Por ejemplo, el analizador Jar acumula datos del manifiesto, pom.xml y los nombres de los paquetes dentro de los archivos JAR analizados. Después de recopilar la información necesaria, emplea un proceso para colocar esta información en uno o más grupos de evidencia.

Se determina la enumeración de plataforma común (CPE) de la dependencia y al resultado se le asigna un nivel de confianza. Este nivel de confianza se basa en la calificación de confianza más baja de la evidencia utilizada. El CPE identificado se registra en el Lucene Index (Índice

Lucene)[8] y posteriormente se coteja con las entradas de Vulnerabilidades y Exposiciones Comunes (CVE)[9] en la Base de Datos Nacional de Vulnerabilidad (NVD)[10], una base de datos de uso gratuito de vulnerabilidades conocidas de seguridad de la información [19].

Dependency-Check se actualiza automáticamente con los datos NVD tan pronto como se ejecuta. Estas actualizaciones garantizan que los informes muestren solo los datos más recientes.

Para habilitar la compatibilidad con lectores de pantalla, pulsa Ctrl+Alt+Z. Para obtener información acerca de las combinaciones de teclas, pulsa Ctrl+barra diagonal.

Los resultados se comparan y están disponibles en varios formatos como HTML, XML, CSV y JSON para que los desarrolladores tomen las medidas adecuadas. Sin embargo, la herramienta no toma el contexto de sus dependencias al informar las puntuaciones de vulnerabilidad. Entonces, los desarrolladores deben verificar si la vulnerabilidad expone su código.

Aquí hay una lista de los analizadores disponibles actualmente como parte de la herramienta, cada uno de los cuales es capaz de escanear un tipo específico de archivo [20]:

- **Archivo**: Extrae y escanea el contenido del archivo
- **Assembly**: Necesita .NET framework o Mono runtime[11]
- **Jar**: Analiza los metadatos del manifiesto del archivo y los archivos del modelo de objetos del proyecto Maven
- **RetireJS**: Examina archivos JavaScript

---

[8] Lucene Index: Un índice de Lucene es una estructura de datos utilizada por el motor de búsqueda Lucene para almacenar y recuperar información de manera eficiente. Este índice se crea a partir de los documentos indexados y se utiliza para acelerar la búsqueda y recuperación de información en un sistema de búsqueda basado en Lucene.

[9] CVE: Es un sistema de catalogación pública que identifica y enumera las vulnerabilidades de seguridad conocidas en productos software y hardware que está desarrollado y mantenido con el respaldo de la comunidad de ciberseguridad.

[10] NVD: La Base de datos nacional de vulnerabilidades es el depósito del gobierno de EE. UU. de datos de gestión de vulnerabilidades basados en estándares representados mediante el Protocolo de automatización de contenidos de seguridad.

[11] ":NET o Mono runtime" es el componente responsable de ejecutar y gestionar la ejecución de estas aplicaciones en un entorno no Windows.

- **Node.js**: Recopila una bill-of-materials[12] después de analizar un paquete.json[13].
- **Node Audit**: requiere acceso a Internet y utiliza API[14] para exponer bibliotecas node.js vulnerables
- **NugetConf**: analiza la especificación XML mediante XPath
- **Nuspec:** Analiza la especificación XML utilizando XPath
- **OpenSSL**: Analiza la definición de macro OPENSSL_VERSION_NUMBER
- **OSS Index**: Similar a Node Audit, utiliza Internet para utilizar API e informar vulnerabilidades que no figuran en el NVD.
- **Ruby Bundler-Audit**: Responsable de ejecutar informes de auditoría de paquetes y
-
- agregar los resultados al informe final.

Además, ofrece varios analizadores experimentales, que aún están en desarrollo y, por lo tanto, podrían generar una gran cantidad de falsos positivos y falsos negativos [15].

- **Autoconf**: Escanea los archivos de configuración del proyecto en busca de metadatos AC_INIT.
- **CMake**: Extrae los comandos de inicialización del proyecto y configuración de la versión
- **CocoaPods**: Escanea el archivo de especificaciones para extraer información de dependencia

[12] Bill-of-materials: Se refiere a un documento que detalla todos los componentes y materiales necesarios para fabricar un producto específico. Esta lista incluye información detallada sobre cada elemento, como su cantidad, descripción, número de parte y proveedor. El objetivo de un bill-of-materials es proporcionar una guía clara y precisa para la fabricación y ensamblaje de un producto.

[13] Paquete.json contiene información sobre el proyecto, como el nombre, la versión, las dependencias, los scripts de inicio, entre otros. Este archivo es fundamental para la gestión de dependencias y la configuración del proyecto en Node.js.

[14] API: Una API (Application Programming Interface) es un conjunto de reglas y protocolos que permite a diferentes aplicaciones comunicarse entre sí.

[15] Falsos positivos y falsos negativos, se refieren a errores o advertencias incorrectas que se generan durante la revisión automática del código. Un falso positivo ocurre cuando el sistema indica que hay un problema en el código que en realidad no existe, mientras que un falso negativo ocurre cuando el sistema no detecta un problema real en el código.

- **Composer Lock**: Extraiga la versión exacta de las dependencias analizando archivos PHP Composer Lock.
- **Lenguaje Go Mod y Dep**: Utilizado en programación para compilar y ejecutar un programa escrito en el lenguaje de programación Go.
- **Python**: Escanea archivos fuente de Python en busca de metadatos de herramientas de configuración
- **Ruby Gemspec**: Escanea los bloques de inicialización de Gemspec en busca de metadatos
- **SWIFT**: Analiza el archivo del paquete Swift.

- *Para su descarga, ver "Presentación, documentación y descarga de OWASP Dependency-Check" en Anexo de instalación (pág. 113).*

2.2.2.3. Ejecución de la herramienta

Para su ejecución, en este caso para el sistema operativo Windows, será abriendo la terminal cmd. En la línea de comando se deberá especificar la ruta de la ubicación del archivo ".bat" que se encuentra en la carpeta de la herramienta descargada previamente, escribiéndola entre comillas, quedando de la siguiente manera.

C:\Users\usuario>"ruta del archivo .bat"

A continuación, al colocar "-h" (help) al final, ejecutando la línea de comando se podrá observar una lista de toda la información para su utilización (Figura 72).

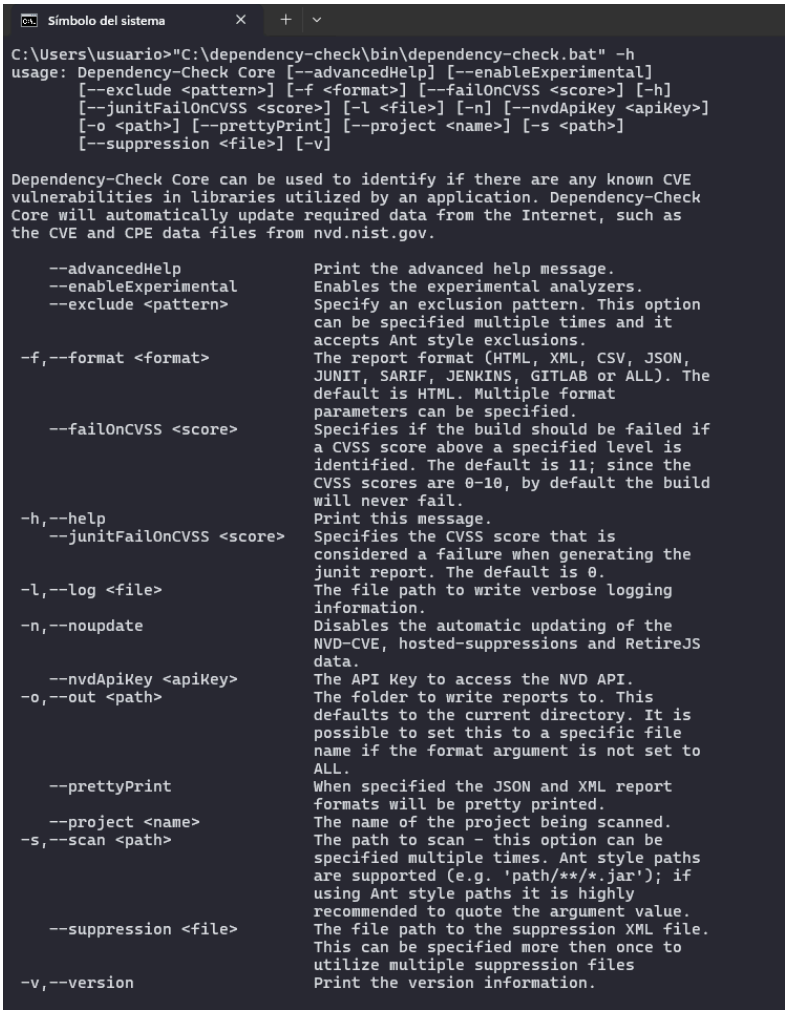


Figura 72: Lista de información para la utilización de OWASP Dependency-Check – Diseño del autor.

Al ejecutar la herramienta por primera vez, descargará automáticamente los archivos de datos CVE y CPE de nvd.nist.gov, esta actualización podrá demorarse dependiendo la conectividad de internet que se dispone (Figura 73).

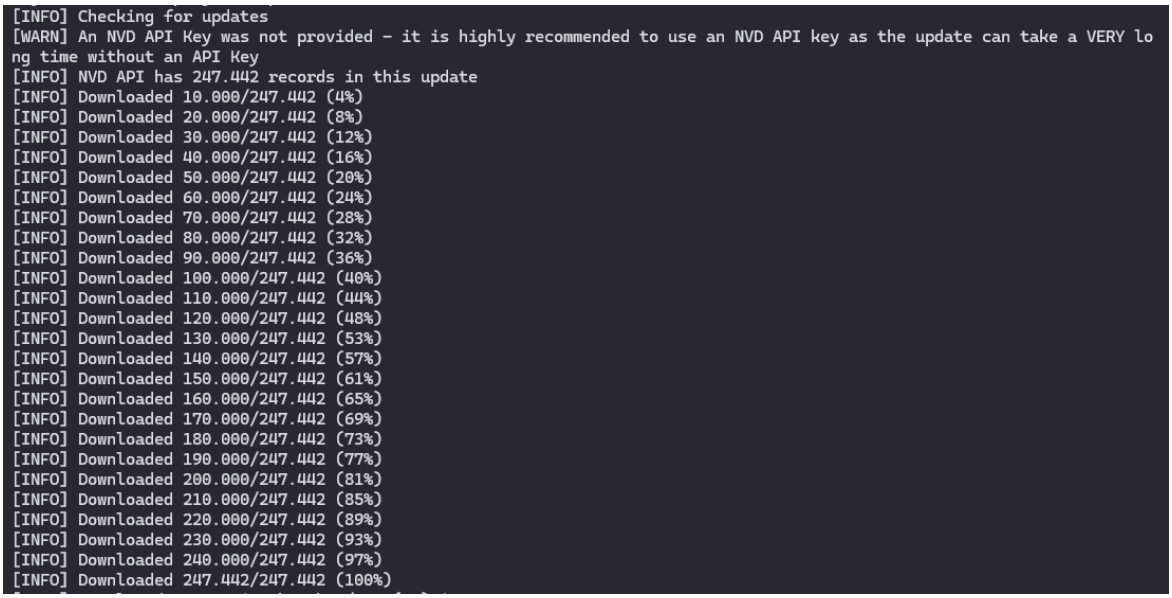


Figura 73: Descarga de actualizaciones de OWASP Dependency-Check – Diseño del autor.

La siguiente tabla muestra la lista de los comandos y su descripción al ejecutar el comando "-h" (Tabla 1).

Comandos	Descripción
`--advancedHelp`	Imprime el mensaje de ayuda avanzada.
`--enableExperimental`	Habilita los analizadores experimentales.
`--exclude <pattern>`	Especifique un patrón de exclusión. Esta opción se puede especificar varias veces y acepta exclusiones de estilo Ant.
`-f, --format <format>`	El formato del informe, puede ser (HTML, XML, CSV, JSON, JUNIT, SARIF, JENKINS, GITLAB o ALL). El valor predeterminado es HTML. Múltiples formatos.
`--failOnCVSS <score>`	Especifica si la compilación debe fallar si se identifica una puntuación CVSS[16] superior a un nivel específico. El valor predeterminado es 11; Dado que las puntuaciones CVSS son 0-10, de forma predeterminada la compilación
`-h, --help`	Imprime esta lista de comandos.
`--junitFailOnCVSS <score>`	Especifica la puntuación CVSS que se considera una falla al generar el informe Junit. El valor predeterminado es 0.

[16] CVSS: Significa Common Vulnerability Scoring System (sistema de puntuación de vulnerabilidad común), es un estándar para evaluar y clasificar la gravedad de las vulnerabilidades de seguridad en sistemas de tecnología.

-l, --log <file>	La ruta del archivo para escribir información de registro detallada.
-n, --noupdate	Deshabilita la actualización automática de NVD-CVE, hosted-suppressions y datos RetireJS[17].
--nvdApiKey <apiKey>	La clave API para acceder a la API NVD.
-o, --out <path>	La carpeta en la que escribir informes. El valor predeterminado es el directorio actual. Es posible establecer esto en un nombre de archivo específico si el argumento de formato no está configurado en TODOS.
--prettyPrint	Cuando se especifican, los formatos de informe JSON y XML se imprimirán bastante.
--project <name>	El nombre del proyecto que se está escaneando.
-s, --scan <path>	La ruta a escanear: esta opción puede ser especificado varias veces. Se admiten rutas de estilo Ant (por ejemplo, 'ruta/**/*.jar'); Si utiliza estas rutas de estilo Ant, se recomienda citar el valor del argumento.
--suppression <file>	La ruta del archivo al archivo XML de supresión. Esto se puede especificar más de una vez para utilizar múltiples archivos de supresión.
-v, --version	Muestra la información de la versión.

Tabla 1. Lista de comandos de ayuda. – Diseño del autor.

Al ingresar el comando “-v”, al ejecutarlo, se podrá observar la versión de la herramienta instalada (Figura 74).

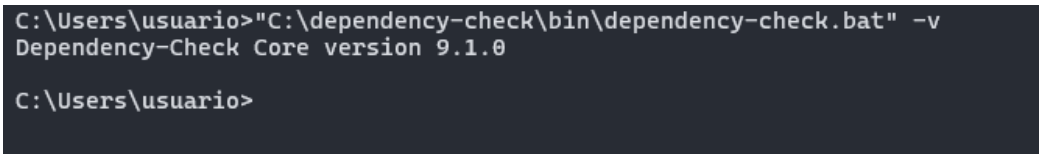


Figura 74: Comando de versión de la herramienta – Diseño del autor.

[17] RetireJS: Es una herramienta de análisis estático que ayuda a identificar bibliotecas de JavaScript obsoletas y vulnerables en un proyecto. Analiza el código en busca de versiones desactualizadas de bibliotecas y alerta al desarrollador sobre posibles problemas de seguridad.

2.2.2.4. Análisis y generador de informes

Para comenzar a analizar un proyecto en busca de vulnerabilidades, se deberá colocar una serie de comandos (Tabla 1) de forma obligatoria para poder realizar dicho análisis. Quedando de la siguiente manera: *"Ruta de archivo .bat" --project Nombre-del-reporte -s "Ruta de la ubicación del proyecto" -o "Ruta donde se guardará el informe"*

Donde principalmente se debe colocar:

- La ruta donde se encuentra la ubicación del archivo .bat de la herramienta (entre comillas).
- Comando "--project" seguido del nombre del proyecto que se reflejará en el informe (con o sin comillas).
- Comando "-s" seguido la ruta donde se encuentra el proyecto o el archivo donde se encuentran las dependencias, según el lenguaje o el tipo de proyecto que se esté analizando. Los cuales pueden ser archivos: .dll, .xml, de lo contrario, se coloca "*" para buscar en todo el proyecto las dependencias (entre comillas).
- Comando "-o" más la ruta de la ubicación donde se guardará el archivo generado (entre comillas) que por defecto es HTML.

Ejemplo:

```
"C:\dependency-check\bin\dependency-check.bat" --project repot-dependencies
-s "C:\project\NodeGoat-master\*" -o "C:\project\report"
```

Al ejecutar la línea de comando con la información, comienzan a funcionar los analizadores en busca de las vulnerabilidades (Figura 75).

Figura 75: Ejecución de análisis de OWASP Dependency-Check – Diseño del autor.

Comienza buscando actualizaciones de la herramienta, su conexión con la API Key de NVD (con sus actualizaciones y base de datos). Además, informa como advertencia, que la herramienta puede dar falsos positivos y falsos negativos. Y por último ejecuta los analizadores disponibles para la búsqueda de las vulnerabilidades.

En la ruta donde se especifica la ubicación donde se va a guardar el reporte, se encuentra un archivo HTML, llamado “dependency-check-report” (Figura 76).

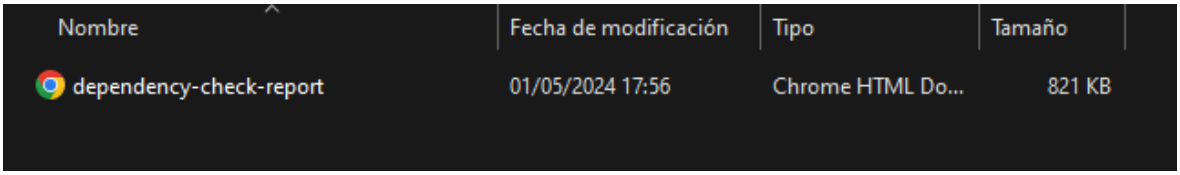
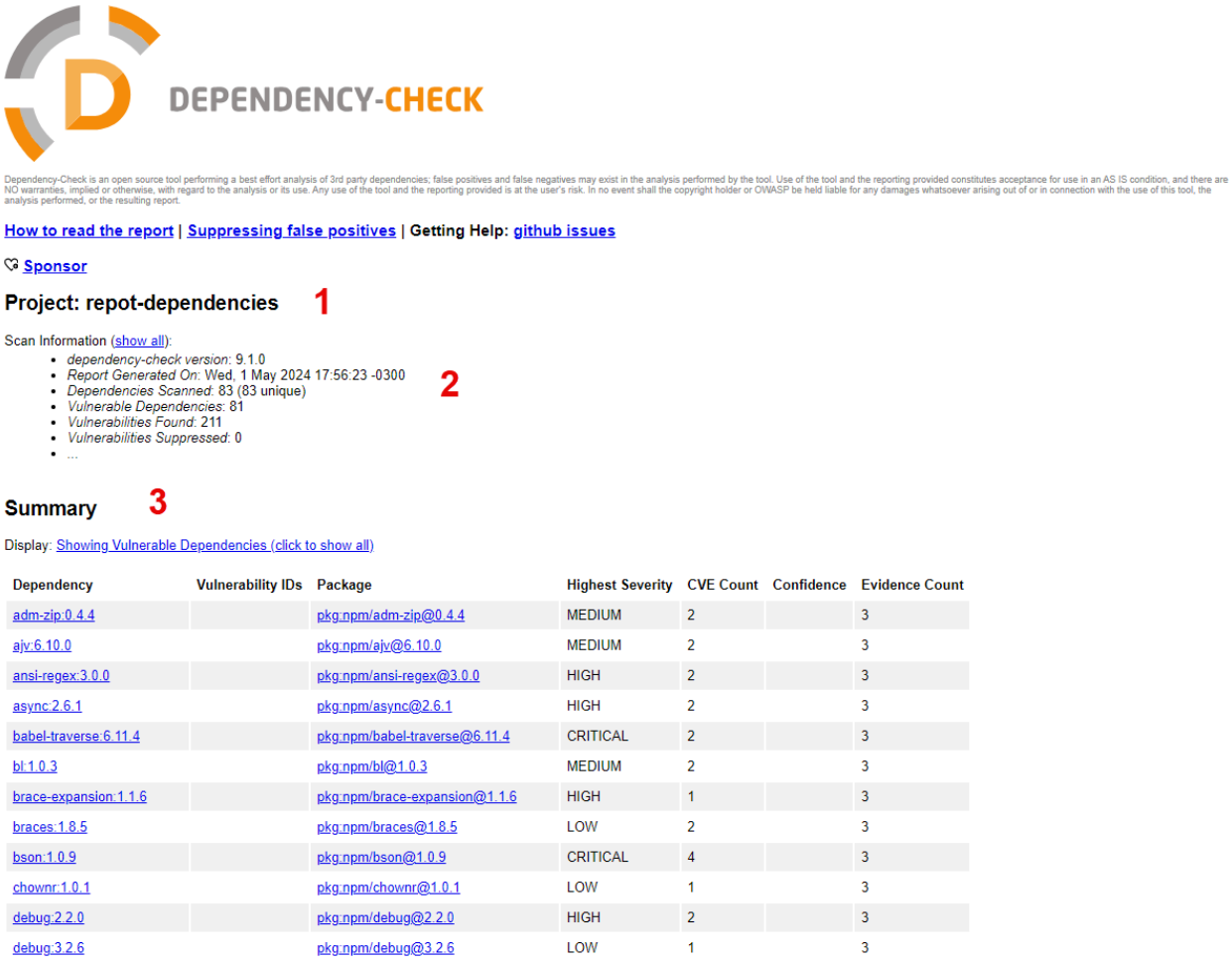


Figura 76: Informe HTML generado por la herramienta – Diseño del autor.

El reporte de OWASP Dependency-Check muestra información detallada sobre las vulnerabilidades potenciales o conocidas de las dependencias de software utilizadas en el proyecto. El propósito de este informe es ayudar a los desarrolladores y equipos de seguridad a

identificar y gestionar riesgos relacionados con las dependencias externas, especialmente bibliotecas y componentes de terceros.

Algunos de los elementos claves que se encuentran al comienzo del reporte son (Figura 77):



DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: repot-dependencies 1

Scan Information (show all):

- *dependency-check version*: 9.1.0
- *Report Generated On*: Wed, 1 May 2024 17:56:23 -0300 2
- *Dependencies Scanned*: 83 (83 unique)
- *Vulnerable Dependencies*: 81
- *Vulnerabilities Found*: 211
- *Vulnerabilities Suppressed*: 0
- ...

Summary 3

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
adm-zip:0.4.4		pkg:npm/adm-zip@0.4.4	MEDIUM	2		3
ajv:6.10.0		pkg:npm/ajv@6.10.0	MEDIUM	2		3
ansi-regex:3.0.0		pkg:npm/ansi-regex@3.0.0	HIGH	2		3
async:2.6.1		pkg:npm/async@2.6.1	HIGH	2		3
babel-traverse:6.11.4		pkg:npm/babel-traverse@6.11.4	CRITICAL	2		3
bl:1.0.3		pkg:npm/bl@1.0.3	MEDIUM	2		3
brace-expansion:1.1.6		pkg:npm/brace-expansion@1.1.6	HIGH	1		3
braces:1.8.5		pkg:npm/braces@1.8.5	LOW	2		3
bson:1.0.9		pkg:npm/bson@1.0.9	CRITICAL	4		3
chownr:1.0.1		pkg:npm/chownr@1.0.1	LOW	1		3
debug:2.2.0		pkg:npm/debug@2.2.0	HIGH	2		3
debug:3.2.6		pkg:npm/debug@3.2.6	LOW	1		3

Figura 77: Reporte de OWASP Dependency-Check – Diseño del autor.

1. **Nombre del reporte**: Muestra el nombre del reporte.
2. **Resumen del Escaneo**: Un resumen general del escaneo, que incluye la versión de la herramienta, la fecha, el número total de dependencias escaneadas, cuántas de ellas tienen vulnerabilidades, y el número total de vulnerabilidades encontradas.

3. **Sumario**: Una lista de todas las dependencias que fueron analizadas, con detalles sobre su versión, su grado de severidad y otras propiedades relevantes.

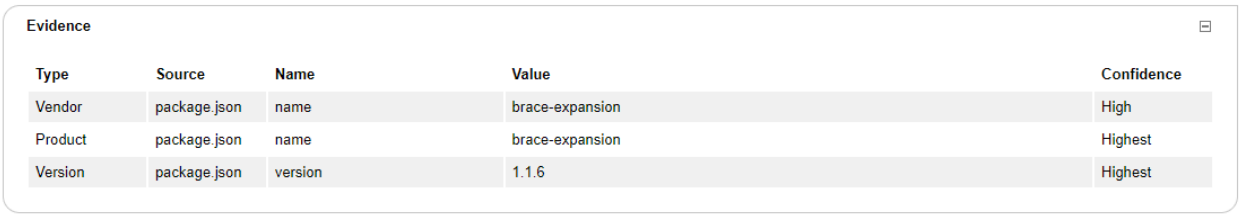
Al hacer clic en alguna dependencia de la lista o al seguir navegando por el reporte, se encontrarán la lista con una descripción más detallada de cada dependencia encontrada con la vulnerabilidad, explicando qué es y cómo puede ser explotada (Figura 78), (Al hacer clic en el símbolo + del costado derecho, se podrá observar con más detalle la información).

Figura 78: Detalle de dependencias en reporte – Diseño del autor.

1. **Nombre, ruta y referencia**: Muestra el nombre de la dependencia con su versión, el archivo Path (la ruta donde se encuentra) y la referencia en el proyecto (Figura 79).

Figura 79: Nombre, archivo Path y referencia de una dependencia en reporte – Diseño del autor.

2. **Evidencia**: Muestra una tabla con detalles de la dependencia encontrada, con su tipo, fuente, nombre, valor y con severidad (Figura 80).

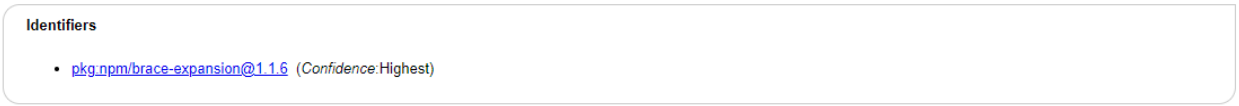


Evidence

Type	Source	Name	Value	Confidence
Vendor	package.json	name	brace-expansion	High
Product	package.json	name	brace-expansion	Highest
Version	package.json	version	1.1.6	Highest

Figura 80: Evidencia de una dependencia en reporte – Diseño de autor.

3. **Identificadores**: Muestra los identificadores de la dependencia (Figura 81).



Identifiers

- pkg:npm/brace-expansion@1.1.6 (*Confidence*:Highest)

Figura 81: Identificadores de una dependencia en reporte – Diseño del autor.

4. **Vulnerabilidades publicadas**: Aquí se encontrará con más detalle sobre la vulnerabilidad sobre la dependencia encontrada. En esta parte, varía la información que muestra dependiendo de las publicaciones realizadas en NVD-CVE, por lo que generalmente muestra el nombre y su enlace de vulnerabilidad, sus recomendaciones (versiones, su métrica en la base de datos CVSS, severidad, referencias, sus versiones de vulnerabilidad, entre otras), (Figura 82).

Published Vulnerabilities

GHSA-832h-xg76-4gv6 (NPM) suppress

Affected versions of `brace-expansion` are vulnerable to a regular expression denial of service condition.

Proof of Concept

```
var expand = require('brace-expansion');
expand('{,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n}');
```

Recommendation

Update to version 1.1.7 or later.

CWE-1333 Inefficient Regular Expression Complexity

CVSSv3:
- Base Score: HIGH (7.5)
- Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Unscored:
- Severity: high

References:
- NPM Advisory reference: - https://bugs.debian.org/862712
- NPM Advisory reference: - https://github.com/advisories/GHSA-832h-xg76-4gv6
- NPM Advisory reference: - https://github.com/juliangruber/brace-expansion/issues/33
- NPM Advisory reference: - https://github.com/juliangruber/brace-expansion/pull/35
- NPM Advisory reference: - https://github.com/juliangruber/brace-expansion/pull/35/commits/b13381281cead487cbdbfd6a69fb097ea5e456c3
- NPM Advisory reference: - https://nvd.nist.gov/vuln/detail/CVE-2017-18077
- NPM Advisory reference: - https://www.npmjs.com/advisories/338

Vulnerable Software & Versions (NPM):

- cpe:2.3:a:*:brace-expansion:\<1.1.7:*:*:*:*:*:*:*

Figura 82: Detalle de la vulnerabilidad publicada en reporte de una dependencia – Diseño del autor.

Además, en el reporte se puede encontrar:

- **Enlaces a Recursos Externos**: Enlaces a recursos externos donde se puede encontrar más información sobre las vulnerabilidades identificadas, como las bases de datos CVE, bases de datos de vulnerabilidades de proveedores, u otros recursos de seguridad.
- **Gráficos y Visualizaciones**: Algunos informes pueden incluir gráficos o visualizaciones que muestran tendencias, como la distribución de vulnerabilidades por severidad, o cuántas dependencias están afectadas por vulnerabilidades.

En el análisis de dependencias se busca identificar componentes de software con vulnerabilidades conocidas. Los resultados de Dependency-Check generan alertas sobre estas dependencias vulnerables. Los cuales pueden ser:

1. **High (Alto)**
    - **Descripción**: Indica que se ha encontrado una vulnerabilidad crítica en una dependencia que podría ser explotada fácilmente y causar un impacto significativo.
    - **Tratamiento**:
        - Actualizar la dependencia a una versión parcheada o segura.
        - Si no hay una versión segura disponible, considerar reemplazar la dependencia con una alternativa segura.
        - Implementar medidas de mitigación temporales, como controles adicionales de seguridad.
2. **Medium (Medio)**
    - **Descripción**: Indica una vulnerabilidad que es significativa pero menos crítica que las de nivel alto. Podría requerir ciertas condiciones para ser explotada.
    - **Tratamiento**:
        - Actualizar a una versión segura si está disponible.
        - Revisar la aplicación para asegurarse de que no se puede explotar la vulnerabilidad bajo las condiciones específicas.
        - Implementar medidas adicionales de seguridad donde sea necesario.
3. **Low (Bajo)**
    - **Descripción**: Indica una vulnerabilidad que es menor y generalmente más difícil de explotar o tiene un impacto limitado.
    - **Tratamiento**:

        - Monitorear la dependencia para actualizaciones futuras que corrigen la vulnerabilidad.
        - Evaluar el riesgo y decidir si es necesario actuar inmediatamente o si se puede diferir la corrección.

4. **Unassigned (No Asignado)**
    - **Descripción**: Indica que se ha encontrado una dependencia vulnerable, pero la severidad no ha sido asignada.
    - **Tratamiento**:
        - Investigar más sobre la vulnerabilidad específica.
        - Tomar decisiones basadas en el contexto y el uso de la dependencia en la aplicación.

Los informes generados por OWASP Dependency-Check están dirigidos a varias audiencias dentro de una organización. Cada audiencia utiliza la información del informe para diferentes propósitos, todos orientados a mejorar la seguridad del software:

- **Desarrolladores**: Identifican y corrigen dependencias vulnerables en el código.
- **Equipos de Seguridad (Security Analysts)**: Evalúan y gestionan la seguridad de las dependencias en el ecosistema de software.
- **Gerentes de Proyecto**: Supervisan el estado del proyecto y asegurar que se abordan las cuestiones de seguridad.
- **Ejecutivos y Alta Dirección**: Aseguran el cumplimiento de las normativas de seguridad y mitigar riesgos potenciales.
- **Compliance Officers (Oficiales de Cumplimiento**): Aseguran el cumplimiento de regulaciones y estándares de seguridad aplicables.

- **Testers**: Verifican que las correcciones de seguridad se han implementado correctamente y que no se han introducido nuevas vulnerabilidades.
- **DevOps y Administradores de Sistemas:** Integran la seguridad en el ciclo de vida del desarrollo y despliegue de software.

2.2.3. Lidiar con los falsos positivos y los falsos negativos

Una vez que los desarrolladores reciben el informe después de la verificación de dependencia, deben verificar si el CPE se identificó correctamente identificando falsos positivos y falsos negativos.

- **Falsos positivos**

Es probable que la metodología de verificación de dependencia genere falsos positivos en los informes, que se pueden suprimir usando el botón de supresión al lado de cada CPE identificado en el informe HTML. Al hacer clic en el botón, puede copiar el XML en un archivo XML de supresión, creando un fichero de supresiones. Esto ayuda a Dependency-Checks a evitar falsos positivos en análisis futuros.

Para crear un fichero de supresiones, en el comienzo del reporte generado, se debe hacer clic en el enlace “Suppressing false positives” (Figura 83).

Figura 83: Enlace de supresión de falsos positivos – Diseño del autor.

Al hacer clic, se redireccionará a la documentación de la herramienta, donde mostrará información de cómo realizar la supresión de dependencias. Para ello, se debe copiar el código XML que brinda la documentación (Figura 84).

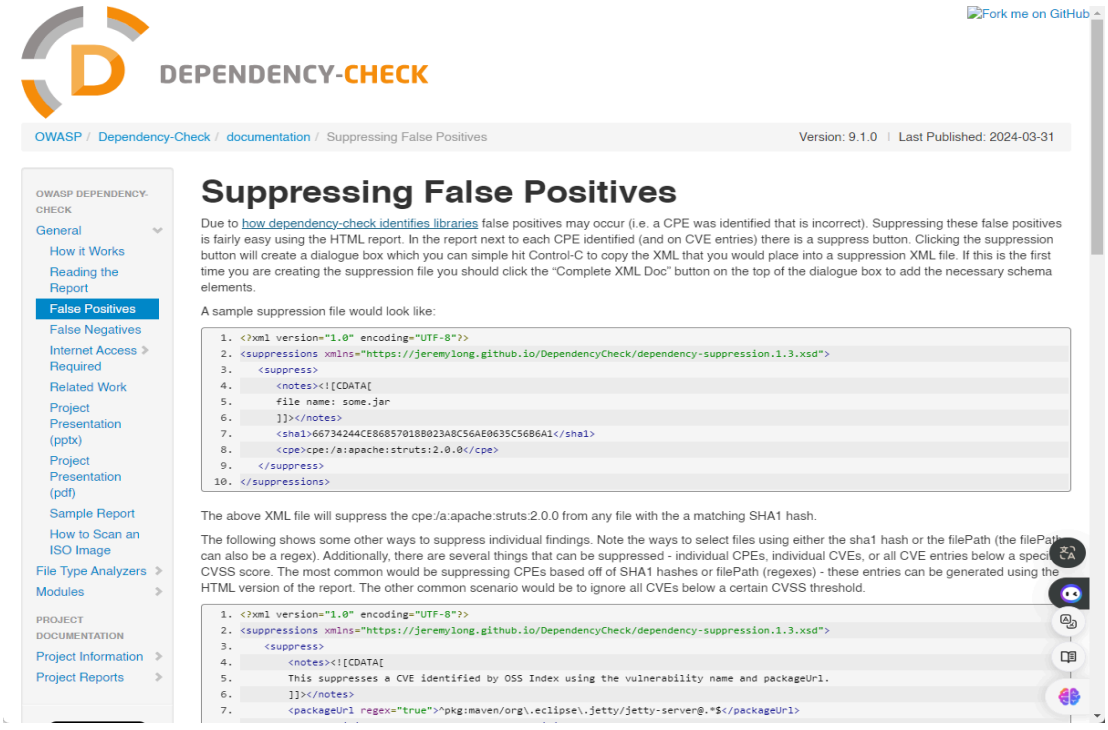


Figura 84: Documentación de supresión de falsos positivos – Diseño del autor.

Una vez copiado el código XML del documento, se crea un Documento de texto (archivo .txt) donde se pegará el código XML copiado (Figura 85).

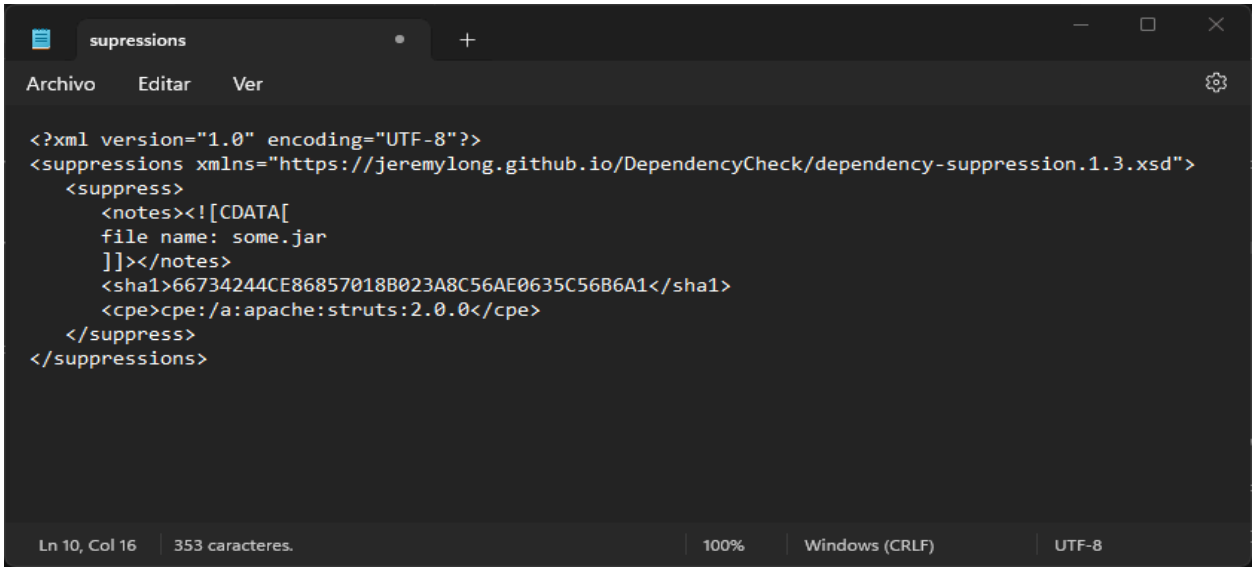


Figura 85: Documento de texto con Código XML para supresión de dependencias – Diseño del autor

Solo queda dirigirse al reporte nuevamente, buscar la dependencia que se desea realizar la supresión, se hace clic en el botón que se encuentra al costado del nombre de la dependencia "suppress" (Figura 86).

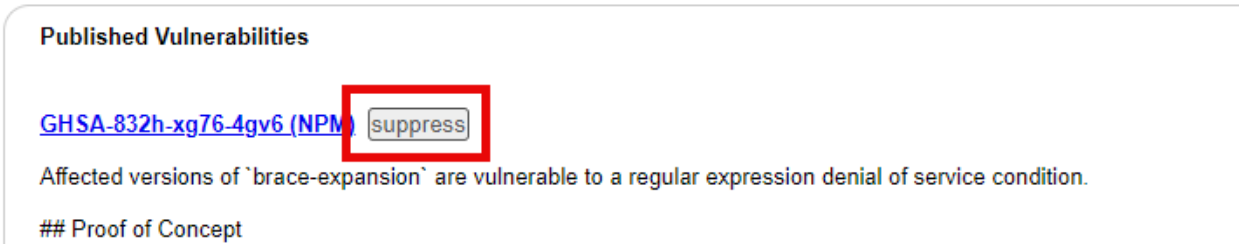


Figura 86: Botón "suppress" para realizar una supresión de una dependencia – Diseño del autor.

Mostrará una ventana con un código XML que se debe copiar y dentro del documento de texto creado anteriormente, pegarlo dentro de las etiquetas "<suppres>" quedando de la siguiente manera (Figura 87).

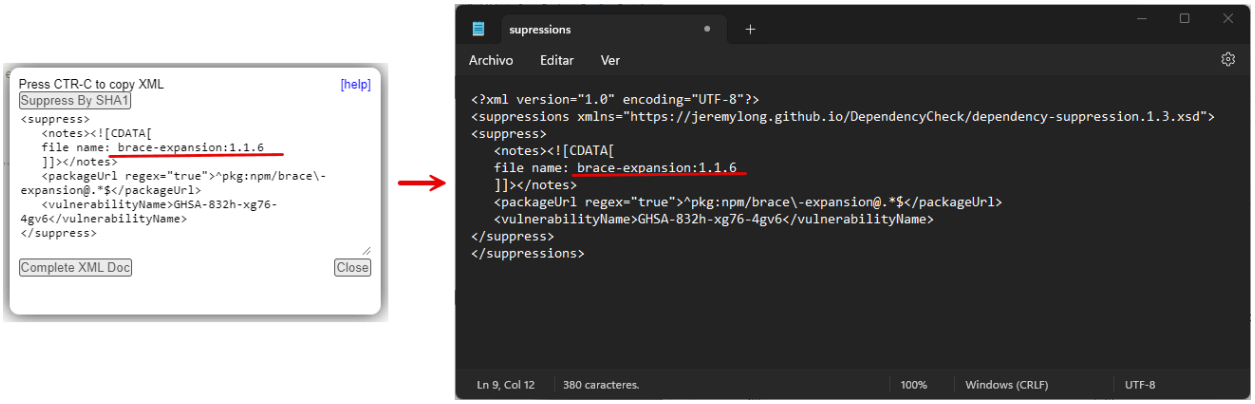


Figura 87: Código XML de dependencia para supresión – Diseño del autor.

Al guardar los cambios del documento, se procede a volver a escanear el proyecto desde la línea de comandos ejecutando la herramienta y se debe agregar el comando "—supression + la ruta del fichero de supresión" quedando de la siguiente manera:

```
"C:\dependency-check\bin\dependency-check.bat" --project repot-dependencies
-s "C:\project\NodeGoat-master\*" -o "C:\project\report" –suppression
"C:\project\report\suppressions.txt"
```

Al comparar el reporte anterior con el generado junto con el fichero de supresión, se puede observar que la dependencia no figura en la lista del sumario (Figura 88).

Figura 88: Comparación de sumarios con y sin fichero de supresión – Diseño del autor.

- **Falsos negativos**

Para identificar falsos negativos, puede utilizar la función 'Mostrar: Mostrar dependencias vulnerables' para revisar las dependencias que no tienen una coincidencia de CPE. Al identificar una dependencia vulnerable sin una coincidencia de CPE, puede crear evidencia para determinar el CPE.

Una vez que los desarrolladores abordan los falsos positivos y los falsos negativos, deben revisar si los CVE identificados son vulnerables a su entorno de software. Sin embargo, una cosa clave a tener en cuenta es que incluso si un CPE se encuentra varias veces, el informe lo incluirá solo una vez. Esto descarga a los desarrolladores la responsabilidad de determinar las ubicaciones donde se necesita mitigar la vulnerabilidad.

Conclusiones y Recomendaciones

Como se ha logrado expresar en el presente trabajo, el entender que la seguridad del software es una necesidad de quien o quienes desarrollan software, es vital para desarrollar y ofrecer software de calidad y esto sin dudas, puede marcar una diferencia entre el éxito y el fracaso de una aplicación, y en el tiempo, el éxito o fracaso de una empresa dedicada al desarrollo de software.

Por lo anterior, el uso conjunto de OWASP ZAP (análisis dinámico) y OWASP Dependency-Check (análisis estático de dependencias) proporciona una cobertura de seguridad más completa, por un lado, posibilita descubrir las vulnerabilidades de seguridad en desarrollos web; por otro lado, lo determina una comunidad mundial, dedicada a los problemas de seguridad de aplicaciones web. Mientras ZAP detecta vulnerabilidades durante la ejecución del sistema, Dependency-Check identifica riesgos en las dependencias externas, permitiendo abordar vulnerabilidades desde diferentes perspectivas y análisis.

Ambas herramientas, permiten detectar una amplia gama de vulnerabilidades conocidas y potenciales, lo que contribuye a una gestión más eficaz de los riesgos de seguridad. Esto ayuda a prevenir brechas de seguridad comunes, como inyecciones, fugas de datos y vulnerabilidades en dependencias de terceros. Es preciso destacar que el análisis estático y dinámico de software, no sólo identifica problemas de seguridad, sino que, también ayuda a mejorar la calidad del software que se produce. Esto es, al detectar prácticas de codificación deficientes y dependencias obsoletas o inseguras, puede resultar la obtención de software más robusto y confiable.

La utilización de herramientas automatizadas de análisis estático y dinámico, además de enumerar las vulnerabilidades encontradas, reportan y sugieren un listado de los posibles

tratamientos que se deberían llevar a cabo, además de indicar el o las áreas involucradas, dentro del proceso de software, para solucionar los inconvenientes reportados por ambos análisis.

Otra de las características estudiadas y analizadas en este trabajo, es que ambas herramientas son altamente automatizables, lo que facilita la integración en flujos de trabajo de CI/CD (Integración Continua/Despliegue Continuo). Esto permite realizar análisis de seguridad de forma regular y escalable, ayudando a identificar problemas a lo largo del ciclo de vida del desarrollo de software. La automatización y el uso de estas herramientas dentro de los pipelines de CI/CD para que el análisis de seguridad se ejecute automáticamente en cada construcción, permite identificar problemas de seguridad rápidamente. Además, esto evita que lleguen a producirse, problemas de seguridad lo que se traduce en demoras y en el no cumplimiento de los tiempos planificados por el equipo de desarrolladores.

En cuanto a las recomendaciones, el integrar OWASP ZAP y OWASP Dependency-Check en el proceso de desarrollo desde el principio, permite establecer puntos de control de seguridad regulares, como parte de las revisiones de código, y antes de cada lanzamiento importante. Para ello, es preciso una capacitación a los desarrolladores y al equipo de QA en el uso de ellas, para luego enfocarse en el análisis, la comprensión e interpretación de los resultados que se obtienen. Lo anterior, brinda una orientación sobre cómo abordar y corregir vulnerabilidades detectadas en las dependencias y planificar actualizaciones o mitigaciones adecuadas, por el equipo de desarrolladores. La adopción de este tipo de herramientas, fomenta la cultura de la seguridad dentro del equipo de desarrollo, alentando al mismo, a considerar la seguridad como parte integral del proceso de desarrollo. A su vez, puede contribuir a una mayor conciencia y a la aplicación de mejores prácticas de seguridad a lo largo del tiempo.

Anexo de instalación

4.1. Instalación de herramienta OWASP ZAP

Para la instalación de OWASP ZAP, se debe descargar la herramienta desde su sitio oficial, donde se encontrará el enlace para su descarga, tanto para Windows, Linux, macOS y los contenedores de Docker, los cuales ya incorporan todo lo necesario para su correcto funcionamiento (Figura 89).

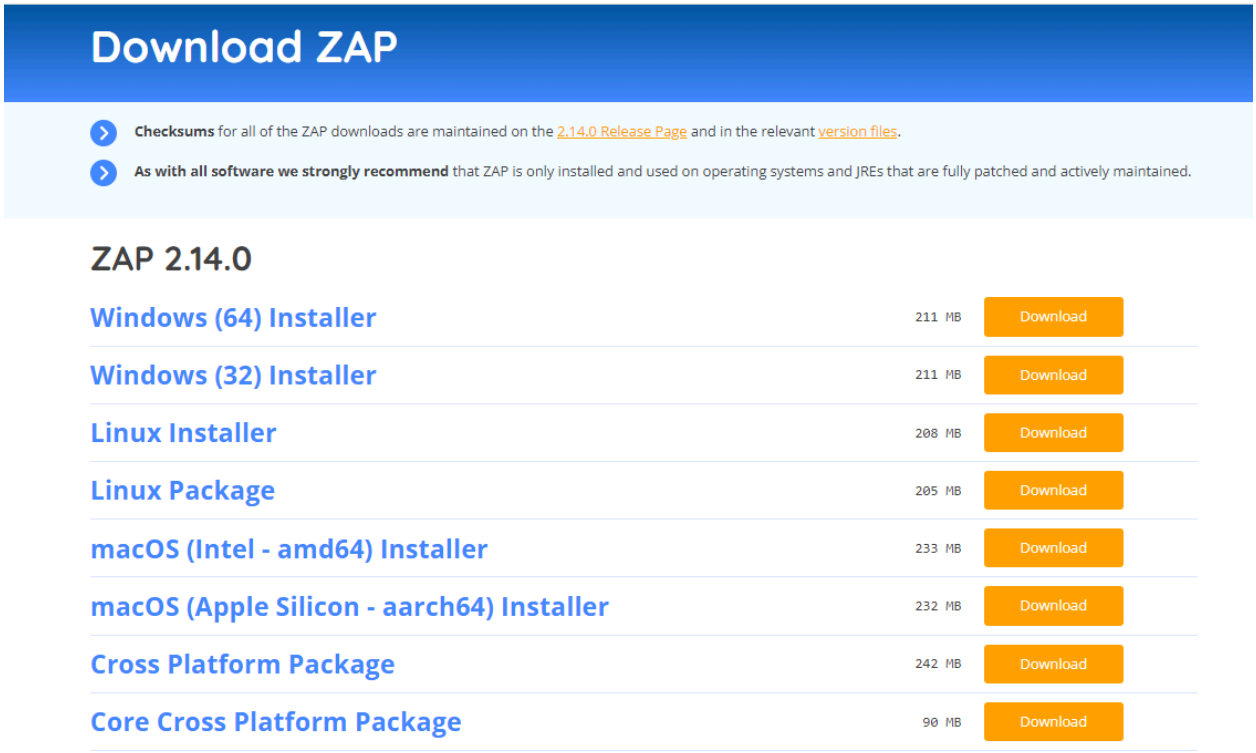


Figura 89: Pantalla de descarga de la herramienta ZAP, OWASP 2023

Es necesario tener instalado Java en el equipo para que ZAP pueda funcionar correctamente. Debe dirigirse a la página oficial de Java y seleccionar la versión que desea instalar.

Una vez descargado el archivo de instalación, la instalación de OWASP ZAP se puede consultar en el anexo 1. Siguiendo el asistente de instalación, mostrará su pantalla de Bienvenida (Figura 90) y se deben aceptar los términos de licencia (Figura 91).

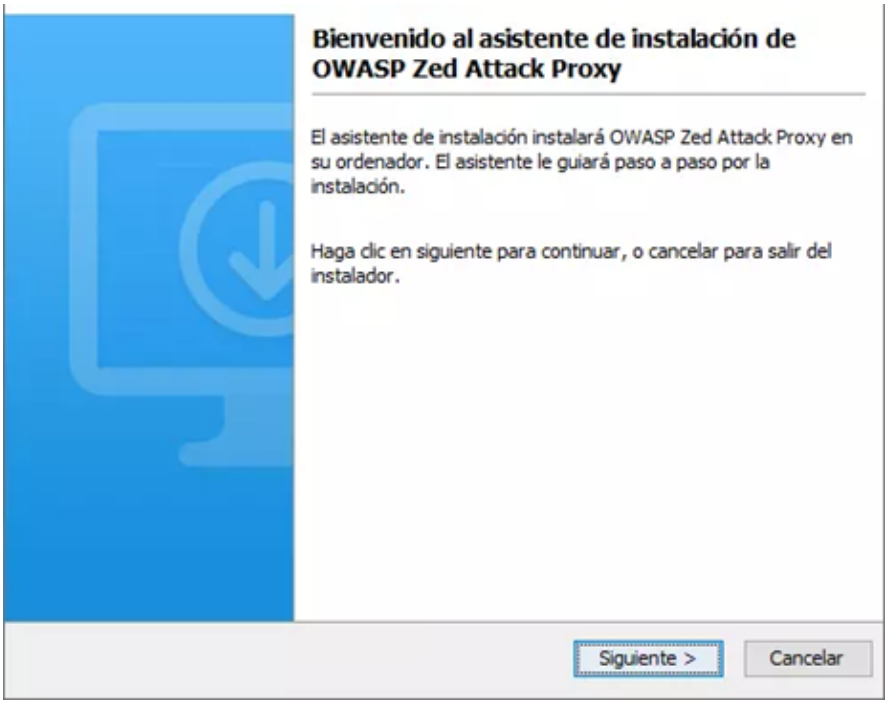


Figura 90: Ventana de Bienvenida del asistente de instalación de OWASP ZAP - Diseño del autor.

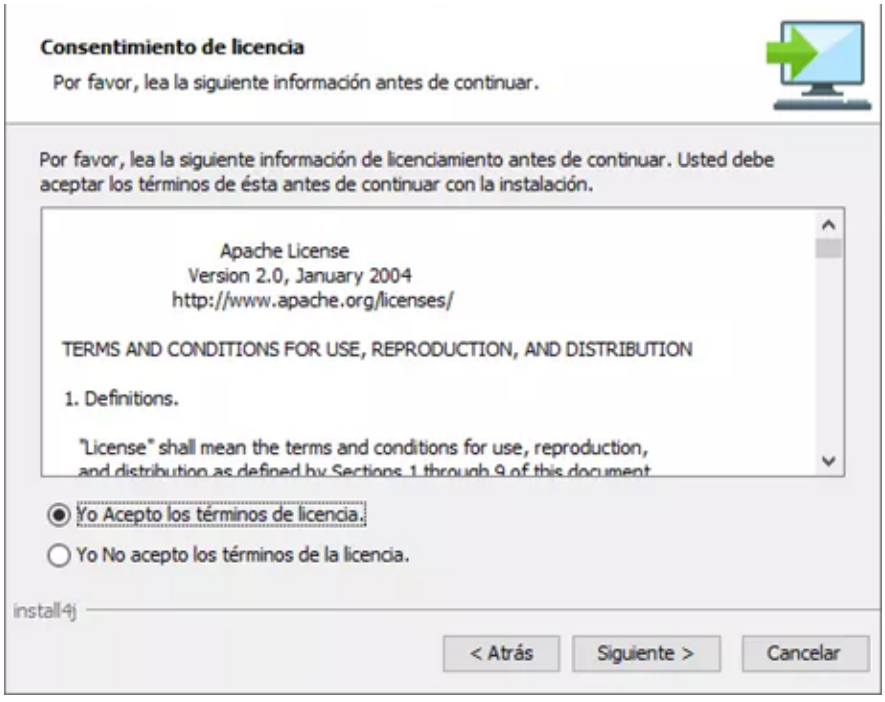


Figura 91: Ventana de Consentimiento de licencia - Diseño del autor.

Luego de la pantalla de consentimiento de la licencia, se debe elegir si se va a realizar una instalación estándar o la personalizada (Figura 92), y elegir el directorio donde instalar todo el programa para ejecutarlo posteriormente (Figura 93).

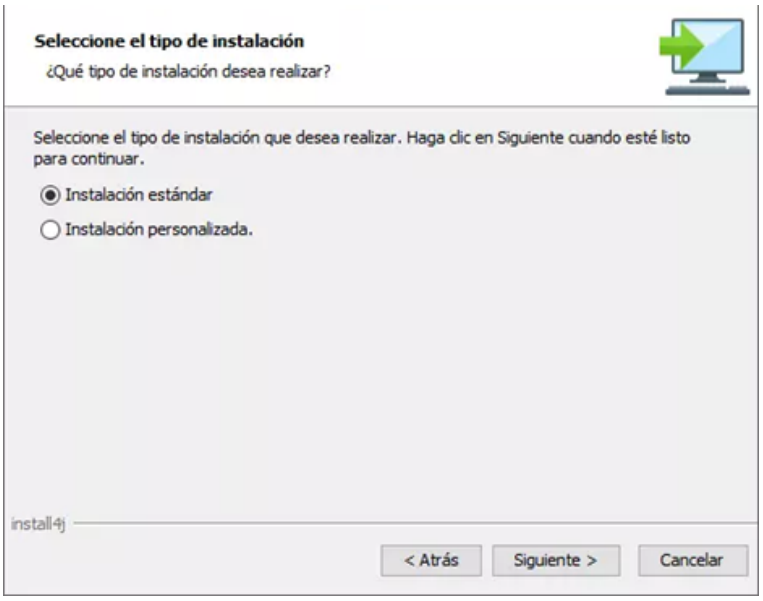


Figura 92: Ventana de selección de tipo de instalación - Diseño del autor.

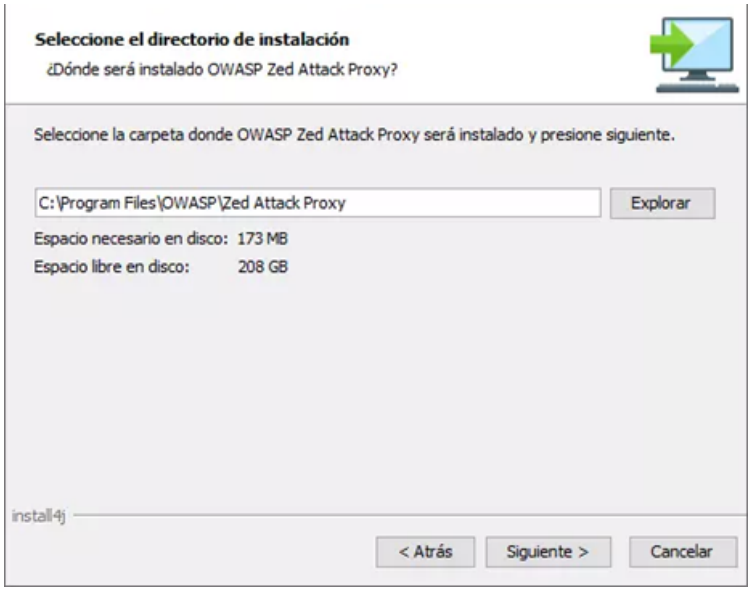


Figura 93: Ventana de selección de directorio de instalación - Diseño del autor.

Seguido de la elección del directorio, se tendrá que elegir si desea crear una carpeta en el menú inicio, crear los iconos en el escritorio e incluso el acceso rápido (Figura 94) y en qué directorio del menú de inicio se crearán las carpetas para los accesos directos de la aplicación (Figura 95).

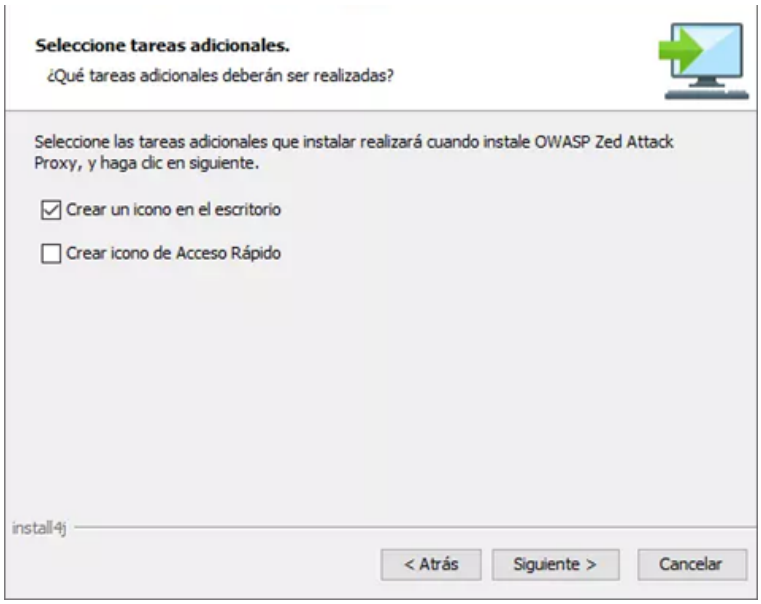


Figura 94: Ventana de selección de tareas adicionales - Diseño del autor.

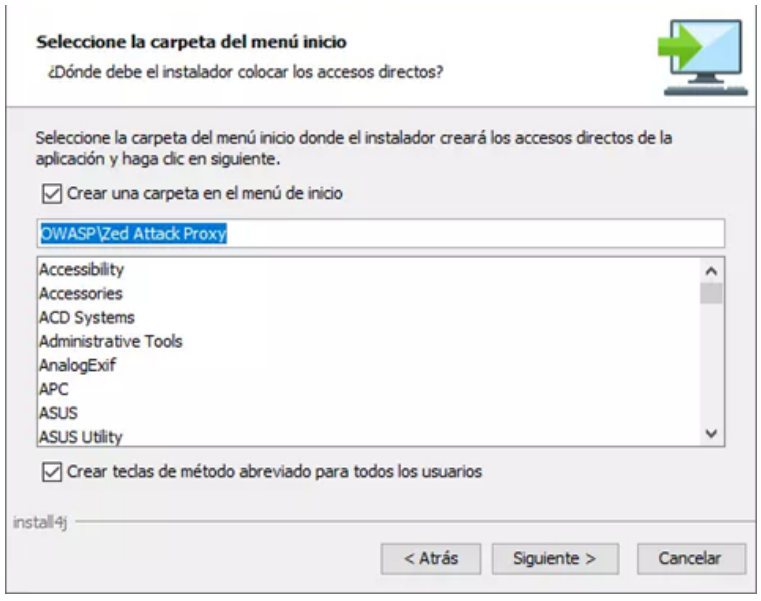


Figura 95: Ventana de selección de la carpeta del menú de inicio - Diseño del auto.

Para finalizar, OWASP ZAP permitirá comprobar si hay actualizaciones. Si es necesario, se podrán instalar las nuevas reglas que se creen para los escáneres web, etc. (Figura 96).

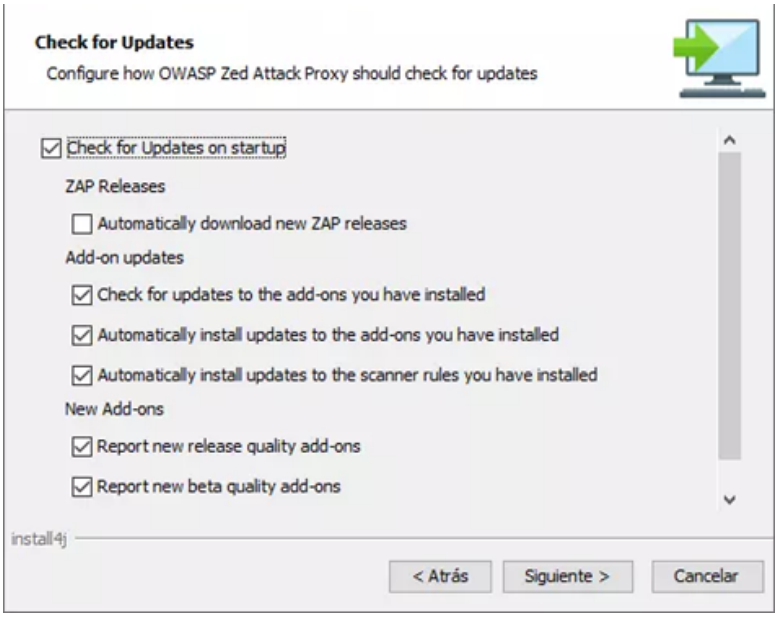


Figura 96: Ventana de chequeo de actualizaciones - Diseño del autor.

Una vez instalado, comienza su ejecución y mientras carga la aplicación, brinda consejos, además realiza un escaneo de actualizaciones (Figura 97).

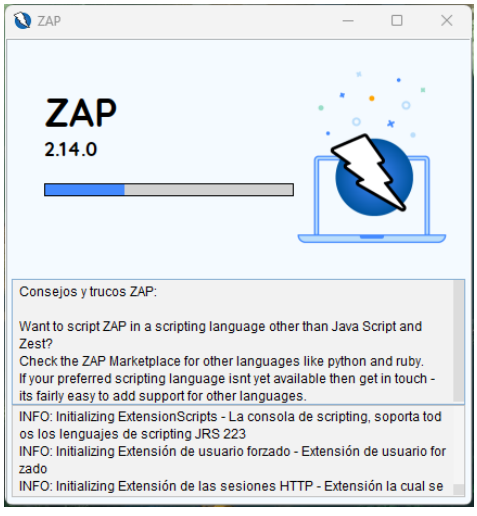


Figura 97: Ventana de carga de la aplicación - Diseño del autor.

4.2. Presentación, documentación y descarga de OWASP Dependency-Check

Al dirigirse al sitio de OWASP Dependency-Check contará con una breve presentación donde explicará el funcionamiento de la herramienta (Figura 98).

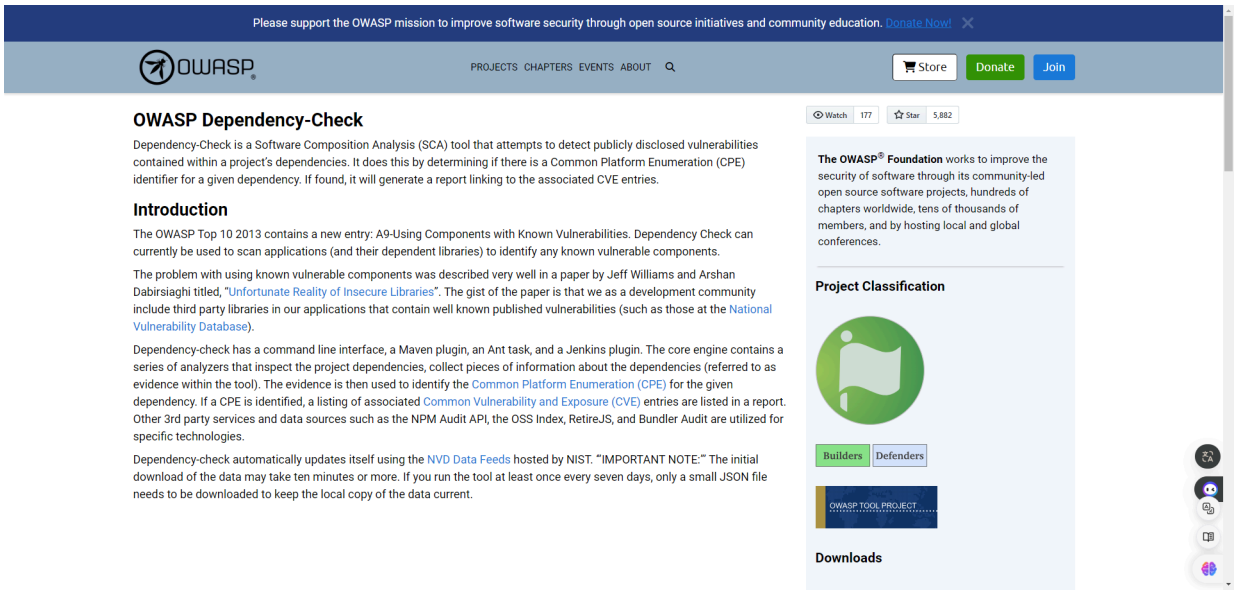


Figura 98: Presentación de OWASP Dependency-Check – Diseño del autor.

Al costado derecho del sitio, al desplazarse hacia abajo, se podrán observar todos los enlaces de su descarga (con su última versión) y sus plugins, sus integraciones, recursos externos, documentación, soporte, entre otras.

En las opciones de descargas, se podrá hacer sin la necesidad de descargarla como integraciones tales como Maven o Jenkins, si no, a través de línea de comando cmd. en la primera opción de la lista, denominada “Command Line” (Figura 99).

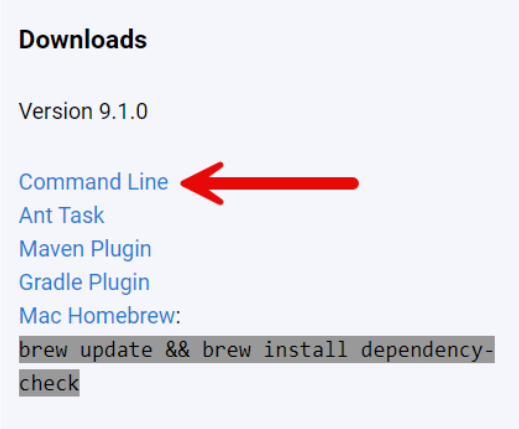


Figura 99: Lista de descargas de OWASP Dependency-Check – Diseño del autor.

Al hacer clic se descarga un fichero Zip (Figura 100), que al descomprimirlo estarán las carpetas con los archivos de la herramienta (Figura 101).

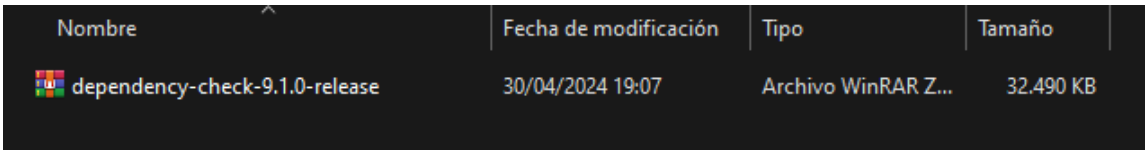


Figura 100: Fichero Zip de la Herramienta – Diseño del autor.

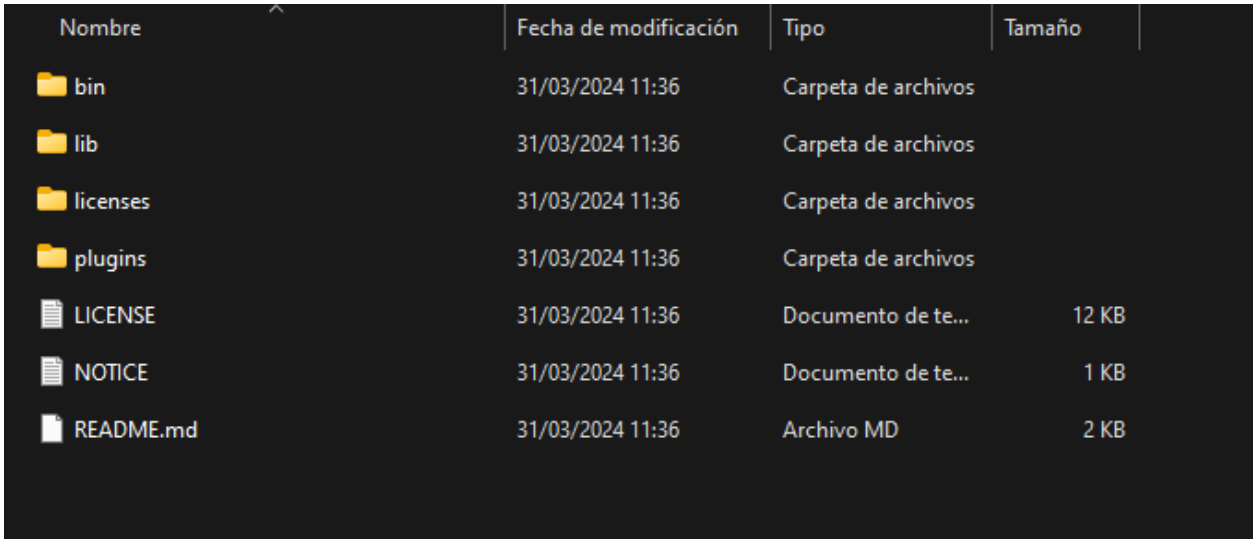


Figura 101: Carpetas y archivos de la herramienta del fichero descomprimido – Diseño del autor.

En el primer directorio “bin” se encontrarán los archivos que se llamarán por línea de comandos. Su ejecución será por medio del archivo “dependency-check.bat” (Figura 102).

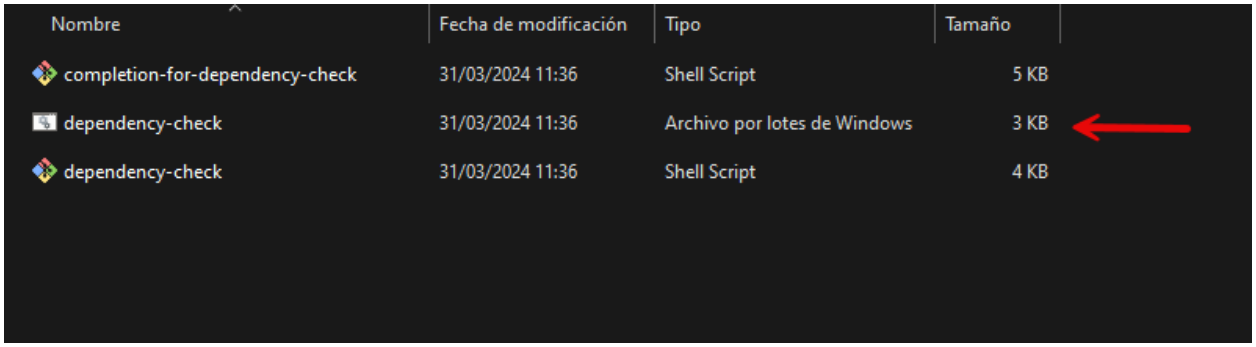


Figura 102: Lista de archivos que se llamarán por medio de comandos – Diseño del autor.

.

Bibliografía

[1] ROBERT C. Seacord (2013) Secure Coding in C and C++ (SEI Series in Software Engineering). Addison-Wesley Professional.

[2] Mark G. Graff y Kenneth R. van Wyk (2003) . Secure Coding: Principles and Practices. O'Reilly Media

[3] Aguilera, P. (2010). Seguridad Informática. Editex. Pozuelo de Alarcón, Madrid. ISBN: 9788497716574.

[4] ACISSI. (2018). Seguridad informática. Hacking Ético. Conocer el ataque para una mejor defensa. Cuarta edición. Editorial ENI. Colección Epsilon. España.

[5] Romero, M. & otros. (2018). Introducción a la seguridad informática y el análisis de vulnerabilidades. Área de innovación y desarrollo, S.L. Primera edición. Alzamora. Alicante. ISBN: 978-84-949306-1-4

[6] William Stallings. (2012). Cryptography and Network Security: Principles and Practices. 6ª Edición. Pearson Education. ISBN: 978-0134444284/ 9780134444283

[7] Roa Buendia, J. F. (2013). Seguridad informática. Editorial McGraw-Hill/Interamericana de España, S. L. España. ISBN: 978-84-481-8569-5.

[8] Marchand-Melsom, A., & Nguyen Mai, D. B. (2020). Automatic repair of OWASP Top 10 security vulnerabilities: A survey. In Proceedings of the IEEE/ACM 4 2nd International Conference on Software Engineering Workshops (pp. 23-30).

[9] Whitman, M. E., & Mattord, H. J. (2016). Principles of Information Security. Cengage Learning.

[10] SERGIO LUJAN MORA. (2002). Programación de aplicaciones web: Historia, principios básicos y clientes web (Spanish Edition). Editorial Club Universitario. isbn: 8484542068

[11]Open Web Application Security Project (OWASP). (s.f.). Recuperado de https://owasp.org/

[12] OWASP Top 10. Recuperado de https://es.wikipedia.org/wiki/OWASP_Top_10

[13] Mark G. Graff y Kenneth R. van Wyk. Secure Coding: Principles and Practices. O'Reilly Media. ISBN: 978-0596002428.

[14] Code Analysis. Recuperado de https://www.codium.ai/

[15] ¿Qué son las herramientas de seguridad DAST, SAST y SCA?. Recuperado de https://codster.io/blog/herramientas-de-seguridad-dast-sast-y-sca/

[16] Análisis estático de código fuente orientado a seguridad: un recorrido teórico práctico. https://www.welivesecurity.com/la-es/2021/01/18/analisis-estatico-codigo-fuente-orientado-a-seguridad/

[17] What is Dynamic Code Analysis?. https://www.checkpoint.com/es/cyber-hub/cloud-security/what-is-dynamic-code-analysis

[18] OWASP. (s.f.). OWASP Zed Attack Proxy Project. https://www.zaproxy.org/

[19] NATIONAL VULNERABILITY DATABASE. https://nvd.nist.gov/vuln/search

[20] How to Analyze the OWASP Dependency-Check? https://www.aquasec.com/cloud-native-academy/supply-chain-security/owasp-dependency-check/