



SEGURIDAD EN SOFTWARE

# HERRAMIENTAS COMO OWASP ZAP

## HERRAMIENTAS COMO OWASP ZAP

**Figura 1.** OWASP ZAP - funciones principales



**Codiga.** (2023, 19 febrero). OWASP ZAP - funciones principales [Imagen]. Codiga.  
<https://www.codiga.io/img/posts/owasp-zap/owasp-zap-core-features.png>

OWASP ZAP, conocida como Zed Attack Proxy, es una herramienta de código abierto diseñada para detectar vulnerabilidades de seguridad en aplicaciones web. Su uso se ha consolidado como una práctica fundamental dentro del análisis automatizado y manual de aplicaciones, especialmente en entornos de desarrollo seguro. Esta herramienta permite interceptar y modificar peticiones HTTP/HTTPS, realizar escaneos activos y pasivos, y generar informes detallados sobre hallazgos críticos como inyecciones, fallos de autenticación o errores de configuración. Gracias a su interfaz intuitiva y su integración con entornos CI/CD, OWASP ZAP se posiciona como un recurso valioso en la implementación de estrategias de seguridad durante el ciclo de vida del software.

### Introducción a OWASP ZAP y su propósito

OWASP ZAP (Zed Attack Proxy) es una herramienta desarrollada por el proyecto OWASP (Open Web Application Security Project) con el propósito de facilitar la identificación de vulnerabilidades de seguridad en aplicaciones web. Está diseñada tanto para profesionales experimentados en pruebas de seguridad como para desarrolladores que desean incorporar prácticas seguras durante el desarrollo de software. Su carácter gratuito y de código abierto la convierte en una opción accesible y ampliamente adoptada en el ámbito de la seguridad informática (OWASP Foundation, 2018).

El propósito principal de OWASP ZAP es actuar como un proxy entre el cliente y el servidor web, permitiendo interceptar, analizar y modificar el tráfico HTTP/HTTPS.

Esta capacidad es esencial para ejecutar pruebas como inyecciones SQL, detección de scripts entre sitios (XSS) o pruebas de autenticación insegura. Su arquitectura modular y extensible permite añadir scripts personalizados y adaptarla a distintos entornos de desarrollo.

Por ejemplo, un equipo de desarrollo que construye una plataforma de comercio electrónico puede utilizar OWASP ZAP para escanear su entorno de prueba. Mediante el escaneo pasivo, ZAP identifica encabezados inseguros y cookies mal configuradas; luego, mediante un escaneo activo, simula ataques para detectar vulnerabilidades más profundas como ejecución remota de comandos. Estos análisis permiten al equipo aplicar correcciones antes de lanzar el producto al público, reduciendo riesgos y fortaleciendo la confianza del usuario final (OWASP Foundation, 2018).

En resumen, OWASP ZAP es una herramienta esencial dentro de cualquier estrategia de seguridad en el desarrollo de software, cuyo propósito es prevenir ataques conocidos, mejorar la calidad del código y proteger los datos de los usuarios.

## Instalación y configuración inicial

La instalación de OWASP ZAP representa el primer paso fundamental para quienes desean realizar pruebas de seguridad en aplicaciones web de forma automatizada y eficiente. Dado que se trata de una herramienta multiplataforma y de código abierto, su implementación es sencilla en sistemas Windows, macOS y Linux.

### 1. Descarga del instalador adecuado

El proceso comienza accediendo al sitio oficial de OWASP ZAP:

<https://www.zaproxy.org/download/>

En esa página, el usuario puede elegir entre múltiples opciones de instalación según el sistema operativo. Por ejemplo:

- En Windows, se recomienda descargar el instalador .exe.
- En Linux, se puede optar por el paquete .tar.gz o utilizar gestores como snap o flatpak.
- En macOS, está disponible una versión .dmg y también puede instalarse mediante brew.
- **Ejemplo:** Una ingeniera de software que utiliza Ubuntu en su estación de trabajo descarga ZAP desde el terminal con el siguiente comando:

**Figura 2.** OWASP ZAP - funciones principales

```
sudo snap install zaproxy
```

## 2. Primera ejecución

Una vez instalado, al iniciar OWASP ZAP por primera vez, el sistema presenta al usuario una ventana de configuración para definir el modo de persistencia de la sesión. Se puede elegir:

- No guardar sesión.
- Crear una nueva sesión.
- Cargar una sesión previa.

Para propósitos de aprendizaje o pruebas temporales, la mayoría de los usuarios opta por no guardar la sesión en el primer uso.

## 3. Configuración de proxy

OWASP ZAP funciona como un proxy local que intercepta el tráfico web. Por defecto, escucha en localhost (127.0.0.1) por el puerto 8080. Para utilizarlo correctamente, el navegador que se use debe ser configurado para enviar el tráfico HTTP/S a través de dicho proxy.

- **Ejemplo:** En Firefox, se accede a las preferencias de red y se establece la dirección del proxy como:

Figura 2. Proxy



Adicionalmente, se recomienda instalar el certificado CA generado por ZAP para evitar advertencias de seguridad cuando se intercepta tráfico HTTPS.

## 4. Interfaz inicial

La interfaz de OWASP ZAP se divide en varias secciones clave:

- **Panel de árbol de sitios:** muestra los dominios y recursos analizados.
- **Pestañas de solicitud y respuesta:** permiten ver el contenido de cada petición HTTP interceptada.
- **Panel inferior:** presenta resultados de escaneos, alertas y registros.

Al iniciar una nueva prueba, el usuario puede ingresar la URL objetivo en la barra superior o dejar que ZAP intercepte el tráfico generado desde el navegador configurado.

## 5. Escaneo rápido

Como paso final de la configuración inicial, se puede realizar un escaneo rápido ingresando una URL. OWASP ZAP analizará la estructura del sitio e identificará posibles vulnerabilidades básicas como encabezados inseguros o formularios sin protección.

### Ejemplo práctico:

Un desarrollador que trabaja en una API REST local puede ingresar `http://localhost:8000/api/` para lanzar un escaneo rápido y verificar posibles brechas antes de su despliegue.

## Escaneo de aplicaciones web

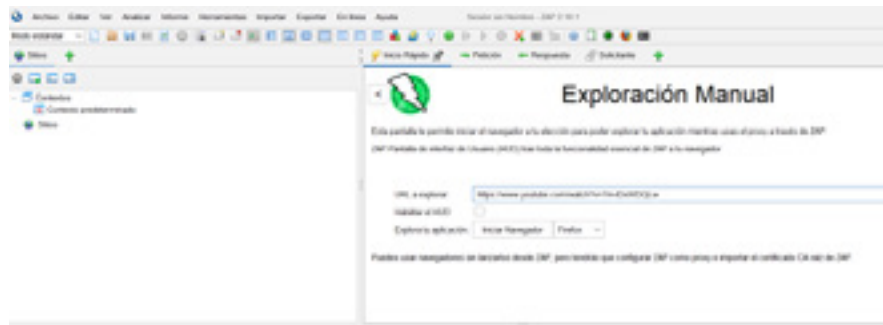
El escaneo de aplicaciones web es una de las funcionalidades más poderosas y utilizadas de OWASP ZAP. A través de esta función, se pueden identificar vulnerabilidades como inyecciones SQL, scripts maliciosos (XSS), configuraciones inseguras y múltiples debilidades en la lógica de una aplicación web. Este proceso, cuando se realiza correctamente, permite descubrir fallos de seguridad de forma temprana, minimizando riesgos antes de llegar a producción (OWASP Foundation, 2018).

### 1. Selección del objetivo

El proceso de escaneo inicia con la definición del objetivo. OWASP ZAP permite ingresar directamente una URL para comenzar el análisis (Ortega Candell, 2018).

- **Ejemplo:** Un tester de seguridad desea analizar una aplicación web en desarrollo alojada localmente. Ingresa en ZAP la dirección `http://localhost:3000/`.

Figura 3. Análisis Web



**Nota.** Creado con OWASP ZAP

### 2. Tipos de escaneo disponibles

OWASP ZAP ofrece diferentes enfoques para realizar el análisis:

- **Escaneo activo (Active Scan):** Simula ataques reales para identificar vulnerabilidades. Es más invasivo.
- **Escaneo pasivo (Passive Scan):** Analiza el tráfico sin realizar interacciones directas que modifiquen el sistema, útil para ambientes sensibles.
- **Escaneo rápido (Quick Start Scan):** Ideal para comenzar con una revisión general automática.



**Figura 4.** Escaneo rápido



**Nota.** Creado con OWASP ZAP

### 3. Ejecución del escaneo

Una vez iniciada la prueba, ZAP examina todas las solicitudes HTTP/HTTPS enviadas y recibidas por la aplicación. Durante este proceso, se construye un mapa del sitio web y se detectan formularios, cookies, parámetros y cabeceras (Ortega Candel, 2018).

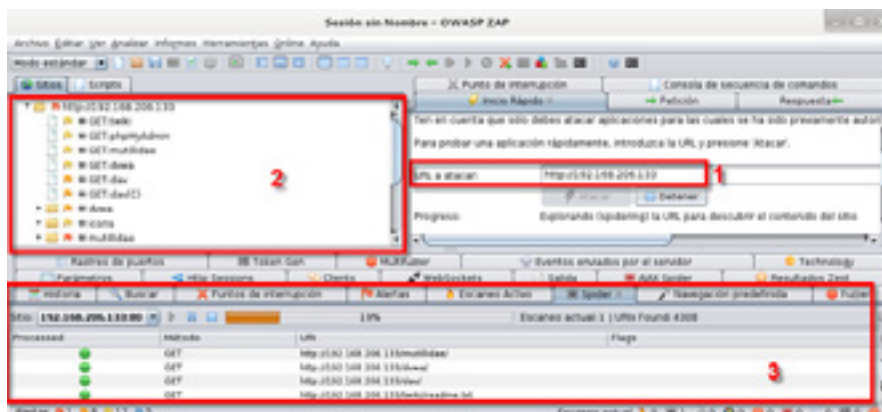
- **Ejemplo:** Durante el escaneo de una aplicación de comercio electrónico, ZAP detecta formularios de búsqueda vulnerables a ataques de inyección de comandos.

### 4. Alertas y resultados

Finalizado el análisis, ZAP muestra una lista de alertas categorizadas por nivel de severidad: alta, media, baja e informativa. Cada alerta incluye una descripción, el riesgo asociado, y una sugerencia de corrección.

- **Ejemplo:** En el análisis de una API pública, OWASP ZAP lanza una alerta por ausencia de cabeceras de seguridad como Content-Security-Policy.

**Figura 5.** Ejemplo de vulnerabilidad



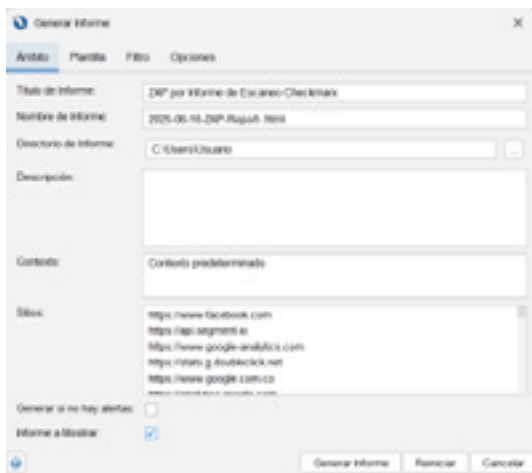
**Nota.** Creado con OWASP ZAP

## 5. Análisis y reporte

Los resultados del escaneo pueden exportarse en formatos como HTML, XML o Markdown para su revisión por el equipo de desarrollo o seguridad. Esto permite integrar los hallazgos con herramientas externas de gestión de incidencias o documentación.

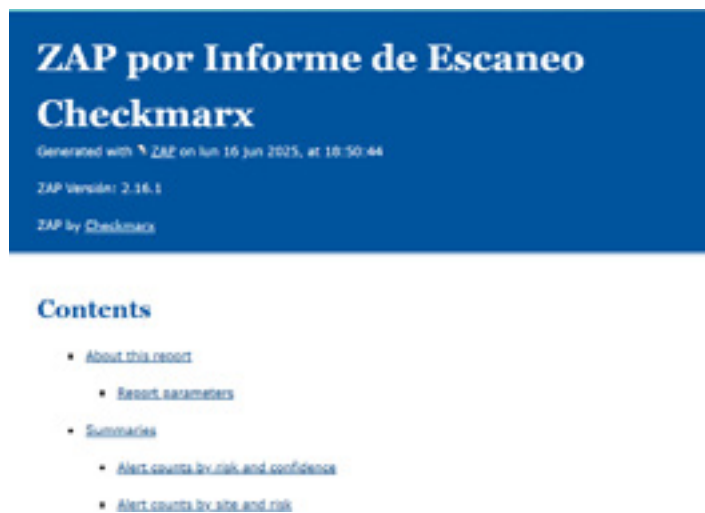
- **Ejemplo:** Un equipo DevSecOps automatiza el escaneo con ZAP y programa la generación diaria de reportes HTML, los cuales se analizan en una reunión semanal de revisión de seguridad.

**Figura 6.** Generación de reporte



**Nota.** Creado con OWASP ZAP

**Figura 7.** Reporte generado por Owasp Zap



**Nota.** Creado con OWASP ZAP

El escaneo de aplicaciones web con OWASP ZAP es una técnica clave dentro de los procesos de aseguramiento de la calidad en ciberseguridad. Esta herramienta permite descubrir de forma temprana riesgos que podrían comprometer la integridad, confidencialidad y disponibilidad de los sistemas. Gracias a su interfaz intuitiva y a su capacidad para realizar escaneos automatizados o dirigidos, se convierte en una aliada fundamental en cualquier flujo de trabajo DevSecOps.

## Uso de herramientas complementarias dentro de ZAP

OWASP ZAP no solo destaca por sus capacidades de escaneo automatizado, sino también por el conjunto de herramientas complementarias que incorpora (Ortega Candel, 2018). Estas funciones adicionales permiten a los analistas de seguridad realizar pruebas más profundas, manipular peticiones HTTP, interceptar tráfico, automatizar tareas e incluso descubrir rutas ocultas o sensibles dentro de una aplicación web. Cada herramienta cumple un propósito específico que, al integrarse en el flujo de análisis, potencia significativamente la detección de vulnerabilidades.

### 1. Spider (rastreador automático)

La herramienta Spider es utilizada para explorar de forma automática los enlaces y formularios disponibles en una aplicación web. Esto permite construir un mapa completo del sitio, identificando rutas dinámicas que podrían pasar desapercibidas en un escaneo manual.

- **Ejemplo:** Al analizar un blog con múltiples categorías, el Spider identifica páginas anidadas que no estaban visibles en la navegación inicial del usuario, permitiendo luego aplicar pruebas de seguridad en esos nuevos puntos de entrada.

### 2. Fuzzer

El Fuzzer de ZAP es una herramienta que permite enviar múltiples cargas (payloads) personalizadas a parámetros de una solicitud, con el objetivo de probar cómo responde la aplicación ante entradas inesperadas o maliciosas. Es ampliamente usado en pruebas de inyección, validación de datos y detección de errores no controlados.

- **Ejemplo:** Un tester de seguridad utiliza el Fuzzer para insertar combinaciones de comandos SQL en el campo “usuario” de un formulario de inicio de sesión, evaluando si el sistema es vulnerable a SQL Injection.

### 3. Interceptador de peticiones (Break tool)

La herramienta Break permite interceptar y modificar en tiempo real las solicitudes HTTP o HTTPS antes de que lleguen al servidor o regresen al cliente. Es ideal para pruebas manuales que requieren alterar parámetros, cookies o cabeceras con precisión.

- **Ejemplo:** Durante el análisis de una aplicación bancaria, el analista modifica el valor de un parámetro `account_id` para verificar si el sistema permite acceso no autorizado a otras cuentas de usuario.

### 4. Zest (scripting visual)

Zest es un lenguaje de scripting visual desarrollado por Mozilla e integrado en ZAP. Esta herramienta permite automatizar escenarios de pruebas complejas sin necesidad de escribir código de programación tradicional. Cada acción se representa como un bloque o nodo visual.

- **Ejemplo:** Un equipo de QA diseña un script en Zest que simula el proceso de autenticación seguido de la edición de un perfil de usuario, integrándose luego en el pipeline de pruebas automatizadas.



## 5. Contextos y autenticación

ZAP permite configurar contextos para manejar distintos entornos de pruebas (por ejemplo: desarrollo, staging, producción) y agregar reglas específicas como autenticación, exclusión de rutas o políticas de acceso.

- **Ejemplo:** Para probar una plataforma educativa protegida por login, el analista configura un contexto con credenciales válidas. ZAP, mediante autenticación automática, puede entonces acceder a áreas privadas del sistema y evaluarlas con el escáner activo.

Las herramientas complementarias de OWASP ZAP ofrecen un abanico de posibilidades para los profesionales de seguridad, permitiendo llevar a cabo análisis tanto automatizados como manuales con mayor profundidad y control. Su uso estratégico, adaptado al contexto y a los objetivos de cada prueba, marca la diferencia entre una simple revisión y un proceso de evaluación de seguridad integral. Integrar estas funciones en el flujo de trabajo garantiza una cobertura más amplia y una detección más precisa de vulnerabilidades en aplicaciones web modernas.

## Generación de reportes y buenas prácticas

La generación de reportes en OWASP ZAP es una fase crítica dentro del proceso de pruebas de seguridad, ya que convierte los hallazgos técnicos en información estructurada y comprensible para los distintos perfiles involucrados en el desarrollo de software. Estos reportes permiten documentar vulnerabilidades, categorizar riesgos, y proponer acciones correctivas, sirviendo como puente entre los equipos técnicos y los responsables de la toma de decisiones (OWASP Foundation, 2018).

### 1. Opciones de reporte disponibles

OWASP ZAP ofrece distintos formatos de exportación que se adaptan a las necesidades de diversos públicos:

- **HTML:** Ideal para presentaciones visuales e informes ejecutivos.
- **XML/JSON:** Útil para integrarse con otras herramientas de análisis o plataformas de seguridad.
- **Markdown:** Atractivo para documentación técnica y wikis internos.
- **PDF (a través de extensiones):** Apto para entregables formales.
- **Ejemplo:** Un analista genera un informe en HTML para compartir con el equipo de desarrollo, mientras exporta la versión JSON para ser consumida automáticamente por un sistema de gestión de vulnerabilidades como DefectDojo.

### 2. Contenido típico de un reporte

Un buen reporte generado por ZAP incluye:

- Fecha y hora de la prueba.
- Alcance del análisis (URLs, dominios y contextos analizados).

- Lista de alertas clasificadas por severidad (alto, medio, bajo, informativo).
- Descripción de cada hallazgo.
- Evidencia técnica (capturas de respuesta, parámetros afectados).
- Recomendaciones específicas para mitigar cada vulnerabilidad.
- **Ejemplo:** El reporte detalla un hallazgo de Cross-Site Scripting (XSS) reflejado, mostrando la URL afectada, el parámetro vulnerable y una recomendación de sanitización de entradas.

### 3. Buenas prácticas al generar reportes

La calidad del reporte determina su utilidad. Algunas buenas prácticas incluyen:

- **Claridad:** Evitar jerga técnica innecesaria cuando el reporte será leído por perfiles no técnicos.
- **Priorización:** Ordenar las vulnerabilidades por nivel de riesgo, permitiendo actuar con mayor eficacia.
- **Evidencia concreta:** Incluir capturas, trazas o payloads utilizados durante la prueba.
- **Contexto de negocio:** Acompañar el hallazgo con su impacto potencial en el negocio o los datos.
- **Trazabilidad:** Relacionar las vulnerabilidades con estándares como OWASP Top 10 o CWE.
- **Ejemplo:** En un entorno financiero, se señala que la falla CSRF en el endpoint / transferencias podría permitir transacciones no autorizadas, afectando la integridad de las operaciones bancarias.

### 4. Automatización de reportes en pipelines CI/CD

ZAP puede integrarse en entornos de Integración Continua para generar reportes de forma automática en cada despliegue o push de código. Esto permite mantener una supervisión continua del estado de seguridad de la aplicación (OWASP Foundation, 2018).

- **Ejemplo:** En un pipeline de GitLab, se ejecuta un escaneo ZAP tras los tests funcionales. El resultado se exporta en JSON y es interpretado automáticamente por una herramienta de control de calidad para frenar el despliegue si se detectan vulnerabilidades críticas.



### 5. Revisión colaborativa de resultados

Una buena práctica es revisar el reporte en conjunto con los equipos de desarrollo, QA y seguridad. Esto permite entender mejor la raíz de los problemas, priorizar tareas y evitar falsas alarmas.

- **Ejemplo:** Durante una reunión de revisión de seguridad, se aclara que una alerta de inclusión de archivos corresponde a una funcionalidad intencional del CMS en modo administrador, marcándose como riesgo.

En conclusión, la generación de reportes en OWASP ZAP no es solo un paso final, sino un instrumento estratégico para comunicar hallazgos de forma clara y accesible. Cuando se aplica con rigor, estructura y enfoque colaborativo, se transforma en una herramienta poderosa para fortalecer la seguridad del software, promover la mejora continua y facilitar decisiones informadas dentro del ciclo de desarrollo seguro.

## Bibliografía

-  Ortega Candel, J. M. (2018). Seguridad en aplicaciones Web Java (ed.). RA-MA Editorial. <https://elibro.net/es/lc/tecnologicadeloriente/titulos/106511>
-  OWASP Foundation. (2018). ZAP 2.9 Getting Started Guide. Open Web Application Security Project. <https://www.zaproxy.org/pdf/ZAPGettingStartedGuide-2.9.pdf>