Análisis de vulnerabilidades en aplicaciones y servicios web utilizando el proceso DAST (Dynamic application security testing)

Jose Giovanni Martinez Cortes

Asesor

Jenny Fernanda Restrepo Santacruz

Universidad Nacional Abierta y a Distancia – UNAD

Escuela de Ciencias, Básicas, Tecnología e Ingeniería ECBTI

Especialización en Seguridad Informática

2026

Dedicatoria

Con mucho amor dedico este trabajo a mi esposa y mis dos hijas quienes han sido mi fuente de inspiración y mi apoyo total para lograr las metas que me he propuesto. Hemos tenido que sacrificar mucho tiempo en familia para que yo pueda cumplir este objetivo. Mi esposa ha asumido roles en el hogar para disminuir mis actividades como padre y permitirme el enfoque en el estudio y trabajo.

Agradecimientos

Estoy muy agradecido con la UNAD en general (Directivos y Docentes) por permitirnos a miles de colombianos acceder a una educación de calidad en la modalidad a distancia y virtual. Gracias al gran trabajo que realizan podemos contar con una plataforma estable, unas tutorías adecuadas y oportunas que permiten que incrementemos nuestro conocimiento de la mejor manera.

Resumen

Esta monografía muestra cómo se pueden encontrar debilidades en la seguridad del software y cómo se gestionan sobre la base del enfoque DAST (Pruebas de Seguridad de Aplicaciones Dinámicas), en el dominio de aplicaciones y servicios web.

Las pruebas de seguridad de aplicaciones dinámicas (DAST) son una técnica de prueba de seguridad de componentes de software que observa la aplicación desde el exterior mientras está en funcionamiento, proporcionando entradas como si estuviera bajo un ataque. La inyección de código, los intentos repetidos de inicio de sesión y las credenciales inválidas son ejemplos de ataques simulados. A diferencia de las tecnologías de inspección de código fuente, como SAST, DAST puede identificar vulnerabilidades que solo son visibles cuando la aplicación está en funcionamiento.

Este documento está destinado a ser una aclaración paso a paso de cómo se realizan las pruebas, las herramientas y técnicas utilizadas, y cómo se resuelven los principales obstáculos para hacerlo. También sirve para demostrar cómo introducir correctamente este tipo de pruebas en el SLDC para garantizar la seguridad de la aplicación desde el principio, en lugar de reaccionar a riesgos inherentes.

El resultado de estas pruebas puede servir como una hoja de ruta para que los desarrolladores o administradores de sistemas mitiguen, remedien y prevengan futuras vulnerabilidades en la aplicación. Esto minimiza el peligro de ciberataques, asegurando que los datos estén seguros y que la integridad no se vea comprometida.

Esta monografía está dirigida a cualquiera que necesite realizar pruebas de seguridad en aplicaciones web y para administradores de sistemas y desarrolladores web que quieran asegurarse de que sus sistemas sean más seguros y confiables.

Palabras clave: DAST, Seguridad, Vulnerabilidades, Ciberseguridad, OWASP.

Abstract

This monograph shows how software security weaknesses can be identified and how these weaknesses are handled based on the DAST (Dynamic Application Security Testing) approach in the domain of web applications and services.

Dynamic Application Security Testing (DAST) is a software component security testing technique that observes the application from the outside while it is running, providing input as if it were under attack. Code injection, repeated login attempts, and invalid credentials are examples of simulated attacks. Unlike source code inspection technologies such as SAST, DAST can identify vulnerabilities that are only visible when the application is running.

This document is intended to be a step-by-step explanation of how testing is performed, the tools and techniques used, and how major obstacles to doing so are resolved. It also serves to demonstrate how to properly introduce this type of testing into SLDC to ensure application security from the outset, rather than reacting to inherent risks.

The results of these tests can serve as a roadmap for developers or system administrators to mitigate, remediate, and prevent future vulnerabilities in the application. This minimizes the risk of cyberattacks, ensuring that data is secure and integrity is not compromised.

This paper is intended for anyone who needs to perform security testing on web applications and for system administrators and web developers who want to ensure their systems are more secure and reliable.

Keywords: DAST, Seguridad, Vulnerabilidades, Ciberseguridad, OWASP.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Introducción

Las aplicaciones y servicios web son herramientas cada vez más utilizadas en el ámbito empresarial y comercial, pero su desarrollo implica enfrentar riesgos relacionados con la seguridad y la calidad del software, estos riesgos se incrementan con la complejidad de las aplicaciones y el tipo de operaciones que se realizan con ellas, por ejemplo, adquisición de productos, servicios, comercio electrónico, pagos en línea, entre otros. Este tipo de operaciones atraen a muchos ciber delincuentes que constantemente buscan vulnerabilidades en componentes de softwares expuestos en internet para poder explotarlas, tener un beneficio monetario y/o generar un daño reputacional a una empresa o marca. Por esta razón, la detección de vulnerabilidades en las aplicaciones y servicios web es un aspecto crítico para garantizar su seguridad, evitar pérdidas monetarias, prevenir un daño reputacional e incluso proteger los datos de los usuarios.

De esta forma, la aplicación del proceso DAST (Dynamic Application Security Testing) se presenta como una alternativa para mejorar la calidad y seguridad en el proceso de construcción de aplicaciones y servicios web. DAST permite detectar vulnerabilidades en tiempo real y brinda información valiosa para la corrección de errores y la optimización del software. Frente a otros procesos como SAST o IAST, DAST puede detectar vulnerabilidades tanto en el aplicativo como en la configuración o infraestructura en la que se encuentra desplegado el producto de software, adicional a esto SAST suele presentar muchos falsos positivos y su tiempo de ejecución es considerablemente amplio frente a DAST. Estas son algunas ventajas por la que DAST debe ser siempre considerado en el momento de analizar vulnerabilidades en un aplicativo o servicio Web. Es de aclarar que para lograr una estrategia efectiva y profunda en el análisis de vulnerabilidades deben combinarse técnicas como SAST, IAST y DAST.

Esta monografía tendrá como pilar ilustrar cómo el proceso DAST puede ayudar de manera significativa en la detección y mitigación de vulnerabilidades existentes en componentes de software, específicamente en aplicaciones y servicios web. Tomará como bases teóricas las metodologías y técnicas existentes en el análisis dinámico de componentes de software.

Definición del Problema

Antecedentes del Problema

Las compañías están apalancándose cada vez más en la tecnología para ofrecer sus productos y servicios. Tomando como base la tecnología construyen productos o componentes de software para permitir a sus clientes autogestionarse, antes que estos productos salgan a producción es necesario que pasen por un análisis de vulnerabilidades riguroso y confiable para mitigar los riesgos de sufrir ataques por personas mal intencionadas que quieran explotar estas vulnerabilidades e impactar de esta forma a la compañía. Los impactos que generen estos ataques pueden causar un daño reputacional, pérdidas millonarias e incluso sanciones debido al incumplimiento en regulaciones, por ejemplo, la exposición de información confidencial de los clientes.

Debido a los argumentos anteriores, es necesario contar con un proceso que recopile las mejores prácticas y métodos existentes para garantizar un análisis de vulnerabilidades muy confiable.

Formulación del Problema

A partir de lo expuesto anteriormente, con el desarrollo de la presente monografía se busca dar respuesta al siguiente interrogante:

¿Cómo la implementación del proceso DAST contribuye a la detección efectiva y precisa de vulnerabilidades en aplicaciones y servicios web antes de su puesta en producción?

Justificación

La construcción de software es cada vez más relevante en la sociedad, las compañías apalancan sus procesos y la obtención de sus objetivos en aplicaciones o componentes de software que sirven de apoyo en las áreas misionales y estratégicas. La disminución de costos de operación, el incremento en ventas y por consiguiente en sus ingresos son apalancados con el uso del software. La construcción, implementación y publicación de servicios para que sus aliados, clientes y proveedores los utilicen son cada vez mayores. La implementación de herramientas de comercio electrónico y otros servicios en Internet se han venido masificando a tal punto que son muy pocas las empresas que de cierta forma no exponen algún componente de software en internet. Lastimosamente, siempre existirán personas malintencionadas que tratan de beneficiarse explotando vulnerabilidades en estos sistemas, aplicaciones o servicios web.

Los Hackers malintencionados o también llamados de Black-Hat buscan constantemente vulnerabilidades en los sistemas que puedan explotar para lucrarse de diferentes formas. Entre estas encontramos: suplantar la identidad de un usuario para acceder a su cuenta bancaria u obtener información sensible para hacer compras con su tarjeta de crédito, acceder a la información sensible de una compañía para causar un daño reputacional, venderla a la competencia o solicitar un dinero para no publicarla, entre otras.

Debido a las anteriores razones, en el ciclo de Desarrollo del Software se han venido implementado desde hace bastante tiempo pruebas de seguridad que permiten detectar vulnerabilidades para que sean mitigadas por los desarrolladores de software o administradores de infraestructura antes de la puesta en producción de los componentes de software. En este trabajo se pretende mostrar cómo se debe planear, analizar y ejecutar un análisis de vulnerabilidades utilizando el proceso DAST en aplicaciones y servicios web.

Objetivos

Objetivo General

Analizar el impacto de la aplicación del proceso DAST en la mejora de la calidad y seguridad durante el desarrollo de aplicaciones y servicios web, a partir de la documentación del proceso, la identificación de vulnerabilidades y la propuesta de un modelo de implementación con sus respectivos reportes de resultados.

Objetivos Específicos

Describir detalladamente el proceso DAST en el contexto de las metodologías de evaluación de aplicaciones y servicios web, incluyendo las tácticas, técnicas y herramientas utilizadas para estructurar un plan de pruebas efectivo.

Identificar vulnerabilidades en aplicaciones y servicios web a partir de la ejecución de pruebas DAST, evaluando la efectividad de las técnicas aplicadas para su detección y verificación.

Proponer un modelo de aplicación del proceso DAST que incluya el diseño del plan de pruebas y los formatos de reporte de resultados, orientado a su implementación y socialización en distintos niveles operativos.

Marco Referencial

Existen dos principales metodologías para realizar pruebas de ciberseguridad sobre aplicaciones o componentes de software. La primera es SAST (Pruebas de seguridad estáticas de aplicaciones) también llamada de caja blanca, el evaluador revisa el código fuente de la aplicación o componente con el objetivo de detectar vulnerabilidades de seguridad. La segunda es DAST (Pruebas de seguridad dinámicas de aplicaciones) también llamadas de caja negra, en esta prueba el evaluador trata de vulnerar la aplicación en ejecución, de la misma forma que lo haría un atacante.

Vamos a profundizar en la metodología que nos interesa en este caso, la metodología DAST se caracteriza por tratar de detectar vulnerabilidades en la aplicación y en la infraestructura sobre la que reposa la misma. En esta metodología la persona que realiza el análisis trata de vulnerar la aplicación tal como lo haría un atacante, de esta forma es menos propensa a reportar falsos positivos que otras metodologías.

Dentro de los beneficios o ventajas de las pruebas DAST podemos encontrar que puede mostrar a la compañía, dueño del componente de software y equipo de desarrollo como se comportaría su aplicación en un entorno productivo ante los ataques que un hacker pueda realizar.

Una de las mayores desventajas de las pruebas DAST es que no indican al equipo de desarrollo la forma de mitigar la vulnerabilidad debido a que no se identifica o revisa el código fuente y esta debe ser mitigada dependiendo del lenguaje de programación e incluso los patrones de diseño implementados.

Marco Teórico

DAST

Según Singh et al. (2024), DAST (Dynamic Application Security Testing) es una técnica utilizada en ciberseguridad para descubrir y evaluar vulnerabilidades de seguridad en una aplicación web en tiempo real. Esta técnica emplea pruebas automatizadas que envían solicitudes a la aplicación web y examinan las respuestas obtenidas para detectar posibles vulnerabilidades, como inyecciones de SQL, cross-site scripting (XSS), inyecciones de comandos y otros tipos de fallos de seguridad (Somi, 2024; Pooniya & Kumar, 2019).

Para ejecutar un análisis DAST existen varias herramientas, de código abierto y propietarias, normalmente pueden evaluarse utilizando algunos criterios como cobertura de vulnerabilidades, tasa de detección, tasa de falsos positivos, tiempo de escaneo y la calidad de los reportes generados. Según Koman y Janiszewski (2025), herramientas como OWASP ZAP, Wapiti, w3af y Codename SCNR son herramientas de código abierto que suelen ser muy efectivas en análisis DAST. A continuación, se muestra un cuadro comparativo con métricas de efectividad propuestas por Koman y Janiszewski (2025).

Tabla 1

Herramientas OWASP y Métricas de Efectividad Propuestas por Koman y Janiszewski (2025)

Herramienta	Cobertura OWASP Top 10	Tasa de detección	Falsos positivos	Tiempo de escaneo	Consumo de recursos	Calidad del reporte
OWASP ZAP	Alta	Alta	Baja	Medio	Medio	Alta
Wapiti	Media	Media	Media	Bajo	Bajo	Media
W3af	Alta	Alta	Media	Alto	Alto	Alta
Codename SCNR	Media	Media	Baja	Medio	Medio	Media

Nota. Adaptado de “SCAnME – scanner comparative analysis and metrics for evaluation”, por J. Koman y M. Janiszewski, 2025, *International Journal of Information Security* (https://doi.org/10.1007/s10207-025-01054-8).

Características de DAST

Utiliza un escaneo dinámico de las aplicaciones web, lo que significa que las pruebas se realizan en tiempo real durante la ejecución de la aplicación (Somi, 2024).

DAST se enfoca en las pruebas de penetración para identificar vulnerabilidades de seguridad en la aplicación web, lo que implica que simula un ataque real para detectar vulnerabilidades en la aplicación (Open Web Application Security Project [OWASP], 2023; Penetration Testing Execution Standard [PTES], 2023).

El análisis DAST se enfoca en la identificación de vulnerabilidades en la aplicación web, como la inyección de SQL, la vulnerabilidad de XSS (cross-site scripting), la inclusión de archivos locales, la vulnerabilidad de CSRF (falsificación de solicitud entre sitios), entre otras.

DAST evalúa la seguridad de la aplicación web y proporciona informes detallados de las vulnerabilidades identificadas (Koman & Janiszewski, 2025).

DAST se puede automatizar para realizar pruebas de seguridad de aplicaciones web de manera continua y en intervalos regulares, especialmente en entornos DevSecOps (Thool & Brown, 2025; López Álvarez, 2020).

DAST es fácil de usar y no requiere habilidades técnicas avanzadas para realizar pruebas de seguridad. Se puede integrar con otras herramientas de seguridad, como análisis estático de seguridad de aplicaciones (SAST), para proporcionar una evaluación de seguridad completa de la aplicación web (López Álvarez, 2020).

Tipos de Pruebas DAST

Escaneo de Vulnerabilidades Automatizado. Este tipo de prueba se enfoca en identificar vulnerabilidades comunes en aplicaciones web, como la inyección de SQL, la vulnerabilidad de XSS, la inclusión de archivos locales, la vulnerabilidad de CSRF, entre otras (Pooniya & Kumar, 2019).

Escaneo de Aplicaciones Web Personalizadas. Tiene como objetivo escanear aplicaciones web personalizadas para identificar vulnerabilidades específicas y personalizadas en la aplicación.

Pruebas de inyección de Datos. Este tipo de prueba DAST simula ataques de inyección de datos para identificar vulnerabilidades en la aplicación, como la inyección de SQL y la inyección de comandos (Singh et al., 2024).

Pruebas de Autenticación y Autorización. Se enfoca en probar los mecanismos de autenticación y autorización de la aplicación web para detectar vulnerabilidades, como la omisión de autenticación y los errores de autorización (OWASP Foundation, 2023).

Pruebas de manejo de errores. Este tipo de prueba se enfoca en probar la capacidad de la aplicación para manejar errores y excepciones de manera segura. Un manejo inadecuado puede exponer información sensible que puede ser utilizada por un atacante.

Pruebas de Escalada de Privilegios. Simula ataques de escalada de privilegios para identificar vulnerabilidades en la aplicación que permiten a los atacantes aumentar su nivel de acceso y control en la aplicación.

Pruebas de Cumplimiento de Seguridad. Este tipo de prueba DAST tiene como objetivo probar si la aplicación web cumple con los estándares y regulaciones de seguridad, como PCI DSS y HIPAA (OWASP Foundation, 2023).

Integrando DAST con el Ciclo de Vida Del Software

Todos conocemos el ciclo de vida del software en el que existe una fase de análisis, continuamos con la fase de diseño, luego los programadores hacen el trabajo de desarrollar el producto, este es probado y si las pruebas son exitosas el producto es liberado en producción para la fase de operación (López Álvarez, 2020).

Se recomienda que las pruebas DAST sean ejecutadas luego de haber culminado la fase de pruebas funcionales, de estrés y carga para garantizar que el artefacto a probar no sea alterado debido a correcciones de errores encontrados y que necesariamente cambien el Código binario generado (Thool & Brown, 2025).

En metodologías ágiles, donde tenemos iteraciones muy cortas, estamos hablando de 1 o 2 semanas, estas pruebas empiezan a ejecutarse luego que las pruebas funcionales superan un 80% de éxito. Esto con el objetivo de no afectar la fecha de paso a producción del artefacto, sin embargo, es bueno aclarar que previo al despliegue en producción debe ejecutarse de nuevo la prueba para garantizar que no existen vulnerabilidades. De existir, estas deben ser mitigadas por el equipo de Desarrollo y se debe aplicar un nuevo test. En metodologías tradicionales como cascada, estas pruebas se ejecutan luego de haber certificado el producto y previo a su puesta en producción.

Según Thool, A., & Brown, C. (2025), en entornos de desarrollo ágil la integración de pipelines de CI/CD con DAST representan desafíos importantes en una organización que al final permiten detectar vulnerabilidades de manera temprana en todo el ciclo de vida del desarrollo de software.

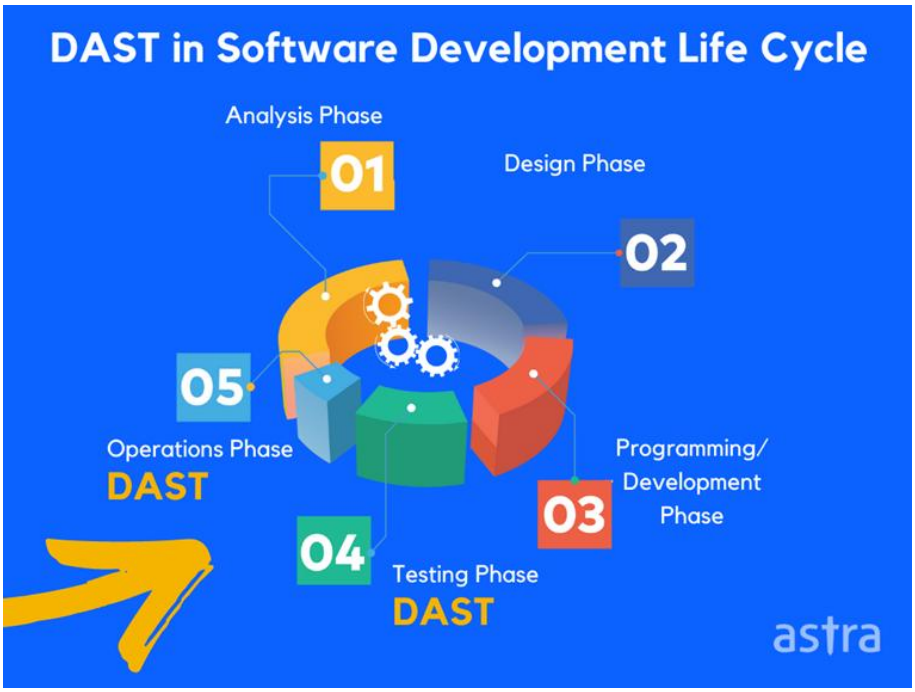
De manera periódica también se deben ejecutar las pruebas DAST en ambiente productivo para detectar vulnerabilidades que puedan haber surgido producto de factores

diferentes al código, estos pueden ser, obsolescencia de protocolos, cambios en el Sistema operativo o incluso despliegue de otros componentes que afectan los componentes ya desplegados y probados (Somi, 2024).

En la siguiente figura se puede evidenciar que las pruebas DAST se deben ejecutar en la fase de pruebas y en la fase de operación del SDLC.

Figura 1

DAST en el Ciclo de Vida del Software



Nota. El diagrama ilustra como interactúa DAST en el ciclo de vida del software. Adaptado de *What is DAST (Dynamic Application Security Testing)? A Complete Guide*, por Astra Security, 2024 (https://www.getastra.com/blog/security-audit/what-is-dast/).

Marco Conceptual

Lanzar un módulo de software, una aplicación o un servicio web en internet o en una intranet es un logro importante para cualquier organización. Por lo tanto, debe ser seguro y debe hacerse correctamente. Los agujeros de seguridad del software deben descubrirse al buscar y corregir vulnerabilidades de seguridad debido a intentos de ataques a la aplicación (Francis, 2025; López Álvarez, 2020).

En este sentido, el enfoque DAST puede descubrir vulnerabilidades en el sentido de que el sistema se prueba como lo haría un atacante con el componente de software realmente implementado. Esto se considera como un miembro externo del equipo o empresa y buscan métodos para comprometer cualquier superficie de ataque que posiblemente se pueda utilizar en su contra (Somi, 2024; Singh et al., 2024).

Ayudar en ese aspecto permitiendo a los desarrolladores descubrir y remediar sus propias configuraciones erróneas es el punto de elevar la calidad y la postura de seguridad con la que se construyen las aplicaciones y servicios web. Durante el proceso DAST, el analista busca debilidades que un humano podría detectar en cualquier punto del ciclo de vida del desarrollo de software. Esto abarca desde la forma en que se desarrolla el software hasta la forma en que se implementa y se apoya. Poder identificar específicamente una debilidad temprano permite corregir esa debilidad, reduciendo así las probabilidades de un ataque exitoso (Thool & Brown, 2025; López Álvarez, 2020).

La integración de DAST en el desarrollo de aplicaciones y servicios web implica pruebas de seguridad que se realizan utilizando herramientas dedicadas que operan contra la aplicación o servicio web. Estas herramientas pueden incluir la búsqueda de vulnerabilidades potenciales

como la inyección SQL, scripting entre sitios (XSS), vulnerabilidades de autenticación y autorización, etc. (Pooniya & Kumar, 2019; OWASP Foundation, 2023).

Garantizar que el proceso DAST se realice de manera que identifique vulnerabilidades que representen debilidades en el código durante la creación de aplicaciones y servicios web es un desafío de incluir la seguridad a lo largo de la codificación de la aplicación y utilizar las herramientas disponibles para encontrar posibles escollos. Esto protege el software y reduce el riesgo de posibles intrusiones (Somi, 2024; República de Colombia, 2009).

Marco Histórico

Las pruebas de seguridad de caja negra han cambiado considerablemente durante las últimas dos décadas. Las pruebas solían ejecutarse a través de ataques cibernéticos manuales que eran altamente técnicos y los analistas requerían muchas habilidades para ser reconocidos y certificados en ciberseguridad. También solo podían informar sobre un número limitado de vulnerabilidades, y su éxito en encontrar vulnerabilidades de seguridad individuales estaba disminuyendo rápidamente (Francis, 2025; Somi, 2024).

Esto impulsó el desarrollo de herramientas de prueba de caja negra más sofisticadas y automatizadas (Francis, 2025; Pooniya & Kumar, 2019). La aparición de los primeros escáneres de vulnerabilidades a principios de la década de 1990 hizo que las pruebas fueran órdenes de magnitud más rápidas y precisas. Estas eran herramientas que podían escanear decenas de millones de líneas de código para encontrar clases de vulnerabilidades en una amplia gama de aplicaciones. (Francis, 2025).

Las pruebas de caja negra han ganado relevancia en los últimos años, con la masificación de Internet y la preocupación por la seguridad y la privacidad. Gracias a lo anterior, muchas organizaciones poseen distintos métodos y formas de encontrar posibles defectos en sus propias

aplicaciones y sistemas, como el análisis automático, auditorías de seguridad y pruebas de penetración (López Álvarez, 2020; OWASP Foundation, 2023).

Las pruebas de caja negra han venido evolucionando a pasos agigantados hasta llegar a las técnicas y herramientas que encontramos hoy en día, por tal razón el futuro es prometedor, la evolución de las aplicaciones tecnológicas es cada vez más impresionante y en ese orden de ideas necesitaremos herramientas y técnicas que nos ayuden a garantizar la seguridad de estas aplicaciones, para lo anterior es fundamental el análisis estático y DAST será el indicado para cumplir con esta tarea (Singh et al., 2024; Thool & Brown, 2025).

Antecedentes o Estado Actual

Según Rini N. (2023), en el año 2010 se origina un concepto denominado DevSecOps debido a la gran necesidad de implementar seguridad en el ciclo de desarrollo del Software. Este término se popularizó en la industria de TI a partir del año 2015 cuando los líderes de la industria comenzaron a hablar de la importancia de integrar seguridad en DevOps. En DevOps muchas tareas se automatizan con el objetivo de facilitar procesos, reducir el tiempo y los riesgos en el ciclo del desarrollo del software, de esta forma se facilita la implementación de DAST en este ciclo y apoyar lo que ahora se llama DevSecOps, colaboración entre desarrolladores, equipo de seguridad y operaciones para construir software de manera ágil, con calidad y seguro (López Álvarez, 2020; Thool & Brown, 2025).

Muchas compañías utilizan DAST y SAST, sin embargo, la automatización de DAST hace que sea el preferido en la industria y aporte un valor significativo en el ciclo DevSecOps (Somi, 2024; Singh et al., 2024). Es de aclarar que existen algunas herramientas como Kiuwan para mejorar la seguridad del código a partir de la implementación de reglas automáticas que

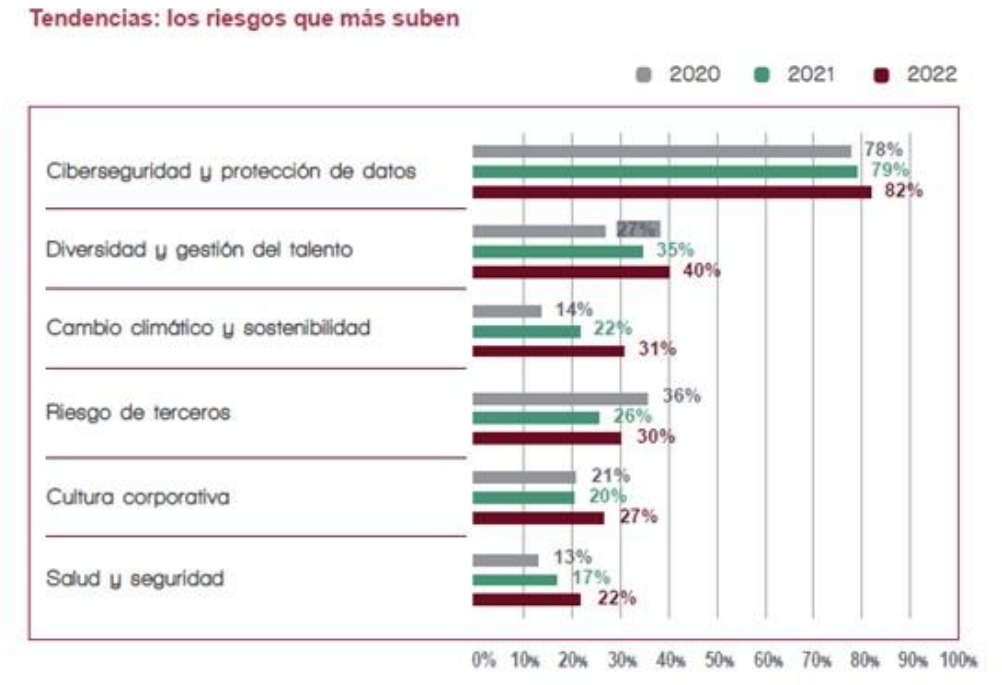
permiten evaluar el código escrito, sin embargo, no es tan efectiva como las herramientas utilizadas en DAST para detectar vulnerabilidades.

En Colombia, solo las grandes empresas tienen un equipo experto dedicado a la ejecución de pruebas de seguridad. Las medianas normalmente tienen proveedores que se encargan de ejecutar análisis por demanda y las pequeñas difícilmente lo ejecutan con alguna periodicidad debido al costo de tener un equipo con el conocimiento necesario República de Colombia (2009).

Debemos concientizar a todos los ingenieros de software de la importancia de construir software seguro y con calidad, el siguiente cuadro muestra los tipos de riesgo que más se han venido incrementando desde el año 2020. Claramente los riesgos de Ciberseguridad se han disparado gracias a la masificación de servicios y aplicaciones en Internet (Francis, 2025; OWASP Foundation, 2023).

Figura 2

Principales Riesgos Empresariales con Mayor Incremento de Relevancia Desde 2020



Nota. El gráfico presenta la evolución de las amenazas críticas, destacando el ascenso de la ciberseguridad y la continuidad del negocio en el entorno europeo. Adaptado de Informe europeo sobre riesgos empresariales: Risk in Focus 2022, por Auditoría-Auditores, 2022 (https://auditoria-auditores.com/articulos/articulo-auditoria-risk-in-focus-2022-informe-europeo-sobre-riesgos-empresariales).

Marco Científico o Tecnológico

Las siguientes etapas describen el plan de acción para tener una buena cobertura de pruebas de caja negra utilizando DAST (OWASP Foundation, 2023; PTES Technical Committee, 2023).

Identificación de los Objetivos de las Pruebas

En esta etapa, se mapean los objetivos de las pruebas, como descubrir problemas de seguridad, probar controles de seguridad o riesgos en el proceso de desarrollo de software (OWASP Foundation, 2023).

Selección de las Técnicas y Herramientas de Pruebas Adecuadas

En esta etapa, se eligen los métodos y herramientas de prueba adecuados para el sistema y los propósitos de la prueba (Somi, 2024).

Planificación y Organización de los Recursos Necesarios

Al planificar pruebas de caja negra, también se debe definir lo que se necesita para realizar las pruebas, incluyendo tiempo, personas, herramientas, hardware, etc. (PTES Technical Committee, 2023).

Ejecución de las Pruebas y Registro de los Resultados

En esta fase, se realizan pruebas de caja negra y se documentan los hallazgos especificando el tipo de vulnerabilidad, la mitigación que se debe tomar y otros comentarios para informar (OWASP Foundation, 2023).

Informe Final

Se requiere documentación para respaldar un hallazgo responsable (informe final) que apoye los resultados de las pruebas, las vulnerabilidades detectadas, las acciones correctivas identificadas y cualquier otra nota pertinente para demostrar la validez de los resultados y cerrar el proceso (PTES Technical Committee, 2023). Ese informe se elabora en esta etapa.

Marco Legal

Según la Super intendencia de industria y comercio (2009), en Colombia, las pruebas de caja negra en ciberseguridad están reguladas por varias leyes y normativas. De acuerdo con la

Ley 1273 de 2009 (República de Colombia, 2009) se establece que cualquier persona que acceda a un sistema informático sin autorización previa puede ser sancionada con pena de prisión y multas SIC 2009 Disponible en: https://www.sic.gov.co/recursos_user/documentos/normatividad/Ley_1273_2009.pdf.

Sin embargo, en el contexto de las pruebas de caja negra, es posible realizar estas pruebas siempre y cuando se cuente con la autorización expresa del dueño del sistema. Esta autorización debe ser otorgada por escrito y debe incluir los límites y alcances de la prueba, así como los resultados que se esperan obtener.

Está complementado por la Resolución 3583 de 2015 (Ministerio de Defensa Nacional, 2015), que establece normas y métodos para realizar pruebas de seguridad informática en sistemas de información estatales y el entorno en el que funcionan. Esta resolución también contempla que el propietario del sistema debe dar aprobación previa y expresa y que la información utilizada en la prueba se confía para mantenerse en confidencialidad. Es crucial considerar la legislación y regulaciones para evitar sanciones legales.

En el sector privado, las empresas y organizaciones a menudo tienen sus propias políticas de pruebas de caja negra para sus sistemas y aplicaciones. Estas directrices suelen indicar cuándo y cómo debe obtenerse el consentimiento previo y expreso, y cuáles son las limitaciones.

Diseño Metodológico

La presente monografía se desarrolló bajo un enfoque mixto, combinando elementos cualitativos y cuantitativos (Hernández Sampieri et al., 2018). Desde la perspectiva cualitativa, está orientado al análisis y comprensión del proceso DAST (Dynamic Application Security Testing) mediante la revisión de documentación científica, estándares y guías especializadas en seguridad de aplicaciones web. Desde la perspectiva cuantitativa, se ejecutaron pruebas dinámicas de seguridad sobre una aplicación real, de tal forma se obtuvieron resultados medibles que permitieron evaluar la efectividad del proceso DAST en un entorno controlado.

El tipo de investigación es descriptiva (Hernández Sampieri et al., 2018) y aplicada (Tamayo & Tamayo, 2011) Es descriptiva, ya que se documenta y explica el proceso DAST, sus etapas, técnicas y herramientas dentro del contexto de las metodologías de evaluación de aplicaciones web. Es aplicada, debido a que los conceptos teóricos revisados fueron llevados a la práctica mediante la realización de un análisis de vulnerabilidades real, con el fin de evaluar el desempeño de DAST.

Como método de investigación se utilizó el análisis documental (Arias, 2012), apoyado en la revisión de artículos científicos obtenidos de bases de datos académicas como Scopus, así como en estándares y guías técnicas relacionadas con la seguridad de aplicaciones web. Este método se complementó con la ejecución de un estudio de caso técnico, consistente en la aplicación del proceso DAST sobre la aplicación OWASP Juice Shop (OWASP Foundation, 2023), utilizando la herramienta OWASP ZAP, herramienta especializada en escanear aplicaciones y servicios web en búsqueda de vulnerabilidades de seguridad.

El diseño metodológico se estructuró en base a las siguientes fases secuenciales.

Fase 1. Fundamentación Teórica y Revisión Documental

En esta fase se realizó la recopilación, análisis y síntesis de literatura científica, estándares internacionales y guías técnicas relacionadas con el proceso DAST, con el fin de establecer el marco conceptual y metodológico del estudio.

Fase 2. Diseño y Configuración del Entorno Experimental

En esta fase se seleccionó la aplicación vulnerable objeto de estudio y se configuró un entorno virtualizado y controlado para la prueba, incluyendo la instalación y la parametrización de las herramientas de análisis dinámico, así como la definición del alcance y las condiciones de ejecución.

Fase 3. Ejecución de las Pruebas Dinámicas y Recolección de Datos

Luego de la configuración del entorno y la instalación de las herramientas necesarias, se procedió con la ejecución del análisis DAST mediante un escaneo automatizado. Luego de pocos minutos, 15 aproximadamente, se obtuvieron los resultados del análisis y se recolectaron datos cuantificables, entre ellos, número de vulnerabilidades detectadas, su nivel de criticidad, tipos de fallas y evidencias técnicas generadas por la herramienta.

Fase 4. Validación y Análisis de Resultados

Los hallazgos obtenidos fueron analizados e interpretados en esta fase. Se verificaron manualmente las vulnerabilidades detectadas con el objetivo de detectar falsos positivos y evaluar la efectividad de DAST.

Fase 5. Diseño del Modelo de Implementación

A partir de la fundamentación teórica y los resultados obtenidos en el caso de estudio, se propuso en esta fase un modelo estructurado para aplicar el proceso DAST, incluyendo el diseño del plan de pruebas y los formatos de reporte de resultados orientados a sus entornos reales.

A continuación, se describe de manera gráfica el procedimiento realizado para construir esta monografía (**ver Figura 3**).

Figura 3

Diseño Metodológico Utilizado en Esta Monografía

Nota. Fases del diseño metodológico utilizado en esta monografía.

Metodologías para Estructurar un Plan de Pruebas en el Proceso DAST

Según Basso y Pretto (2019), existen varias metodologías, frameworks y/o estándares para estructurar un enfoque de prueba DAST en aplicaciones y servicios web. Estas pruebas son utilizadas para detectar fallas y vulnerabilidades de seguridad 'interactuando' con la aplicación en tiempo de ejecución (Somi, 2024; Singh et al., 2024; OWASP Foundation, 2023). Los pasos son básicamente: definir los requisitos de la prueba, identificar el objetivo, formular los escenarios de prueba, configurar el entorno de pruebas, ejecutar la prueba, analizar los resultados, presentar el informe ejecutivo y operacional (OWASP Foundation, 2023; Penetration Testing Execution Standard, 2023). Utilizando estos métodos, las empresas incrementan la seguridad en sus aplicaciones y servicios web al identificar y mitigar vulnerabilidades antes de que estas sean explotadas por atacantes malintencionados. A continuación, se describen las metodologías más relevantes.

OWASP

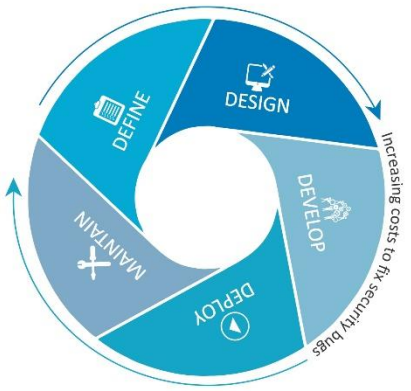
La Open Web Application Security Project (OWASP) es una organización sin ánimo de lucro que se dedica a mejorar la seguridad de las aplicaciones web. La metodología OWASP se basa en una lista de los 10 riesgos de seguridad más críticos para las aplicaciones web. La metodología proporciona una guía detallada para la realización de pruebas de seguridad en las aplicaciones web. La última versión mayor de la guía (OWASP 4) fue liberada en el año 2014, en el año 2020 se liberó la última versión menor (OWASP 4.3) disponible en el sitio oficial https://owasp.org/www-project-web-security-testing-guide/ (OWASP Foundation, 2023).

OWASP recomienda tener en cuenta aspectos de seguridad en cada una de las fases del desarrollo de software, claramente entre más temprano se empiecen a tener en cuenta esos aspectos, menos será el costo de corregir los defectos de seguridad (López Álvarez, 2020; Thool

& Brown, 2025). A continuación, se muestra una figura que plasma las etapas que tiene el ciclo de vida de desarrollo del software.

Figura 4

Ciclo de Vida del Desarrollo de Una Aplicación



Nota. El diagrama ilustra lassss fases del ciclo del desarrollo del software. Adaptado de *Web Security Testing Guide (WSTG) v4.2*, por OWASP Foundation, s.f. (https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/README).

Fases de un Plan de Pruebas Según OWASP

Según la metodología estipulada por la OWASP Foundation (2023), la ejecución de un análisis dinámico requiere una estructuración secuencial que trascienda el simple acoplamiento de herramientas automatizadas. Un plan de pruebas DAST efectivo demanda un orden cronológico y lógico de fases orientadas a maximizar la cobertura de la superficie de ataque; este orden mitiga el riesgo de obtener falsos positivos y garantiza la integridad de los entornos evaluados. Las etapas que componen este flujo se detallan a continuación.

Identificar los Objetivos. Los objetivos de interés que serán explorados mediante pruebas de seguridad deben ser establecidos durante la definición de objetivos. Estos objetivos deben ser cuantificables, concretos y relevantes para el proyecto.

Identificar las Amenazas Potenciales. Se identifican las clases relevantes de amenazas que son abordadas por la seguridad. Para esto, se puede utilizar la lista proporcionada por OWASP de amenazas estándar, que son:

- Inyección de código malicioso
- Autenticación y gestión de sesiones
- Exposición de datos sensibles
- Entidades externas XML XXE
- Control de acceso
- Vulnerabilidades en la configuración
- Cross-Site-Scripting (XSS)
- Seguridad insuficiente en el transporte
- Deserialización insegura
- Utilización de componentes con vulnerabilidades conocidas

Definir los Casos de Prueba. De acuerdo con las amenazas generadas, se deben definir los casos de prueba para validar la seguridad de la aplicación. Estos casos de prueba deben ser demostrables, medibles y vinculados a los objetivos de seguridad ya establecidos.

Identificar los Requerimientos de Prueba. Se deben evaluar las necesidades requeridas para las pruebas, incluidas las pruebas de seguridad. Esto implica personal, hardware, software y tiempo.

Definir los Criterios de Aceptación. Se establecen criterios de aceptación para medir la calidad del reporte de las pruebas de seguridad. Estos criterios deben derivarse de los requisitos de seguridad especificados y los casos de prueba.

Definir la Estrategia de Pruebas. Se determina una estrategia de pruebas para llevar a cabo pruebas de seguridad. Entre estas se encuentran la naturaleza de las pruebas, el alcance de las pruebas y la frecuencia de las pruebas.

Planificar las Pruebas. La planificación de las pruebas implica la firma de los requisitos de recursos, la definición del cronograma de pruebas y la definición de los roles y responsabilidades de los miembros del equipo.

Ejecutar las Pruebas. Según lo programado, se realizan las pruebas de seguridad. Durante las pruebas, se debe registrar y documentar lo que sucede.

Analizar los Resultados. Es necesario analizar los resultados de las pruebas de seguridad para encontrar vulnerabilidades, riesgos de seguridad o debilidades. Se debe evaluar el nivel de riesgo para cada vulnerabilidad y su forma de mitigación. En este análisis se puede evaluar el impacto que puede generar la vulnerabilidad e incluso concluir si es un falso positivo. Este último se puede determinar mediante una prueba manual tratando de explotar la vulnerabilidad como lo haría un atacante.

Reportar los Resultados. En esta fase crea y socializa un reporte con los resultados de las pruebas de seguridad a los miembros del equipo responsables de la aplicación. El informe debe incluir un resumen de las vulnerabilidades y los riesgos identificados, así como recomendaciones para abordarlos y/o mitigarlos. En el reporte siempre se hace referencia a las recomendaciones de mitigación dadas por el mismo OWASP. En estas referencias podemos encontrar recomendaciones para los lenguajes de programación más utilizados, entre estos, C#, Java, PHP, python, entre otros.

PTES

El Estándar de Ejecución de Pruebas de Penetración (PTES) puede ser utilizado para evaluar la seguridad de una aplicación o servicio de internet. PTES es un enfoque estructurado excelente y contiene los detalles precisos de una prueba de penetración exitosa. Las directrices técnicas para tales pruebas de penetración se pueden encontrar en este enlace: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

Este importante estándar contiene 7 etapas para cubrir el ciclo completo de pruebas de penetración.

Fases de un Plan de Pruebas Según PTES (Penetration Testing Execution Standard)

Según el PTES Technical Committee (s. f.), las siguientes son las fases y el orden que debe tenerse en cuenta para realizar un análisis efectivo utilizando el método DAST (Penetration Testing Execution Standard, 2023; Somi, 2024).

Preparación. Incluye determinar el propósito de la prueba, decidir el alcance de la prueba, decidir los módulos/funciones de software a probar, identificar los riesgos potenciales, determinación del presupuesto y el personal que llevará a cabo la prueba (OWASP Foundation, 2023; López Álvarez, 2020).

Recopilación de Información. Se obtiene toda la información requerida para los componentes a probar. Tal conocimiento puede incluir sistemas, aplicaciones, servicios, direcciones IP, puertos, protocolos, nombres de dominio, topologías de red, descripciones de red, políticas de seguridad, usuarios y grupos, etc. (Singh et al., 2024).

Análisis de Vulnerabilidades. La intención de este elemento de seguimiento es identificar todas las vulnerabilidades o debilidades de los componentes de software que comprenden el alcance de la prueba. Esta relación es calificada por escáneres de vulnerabilidades (Pooniya & Kumar, 2019; Koman & Janiszewski, 2025).

Explotación. La explotación es el momento en el que las amenazas que se entienden y documentan son explotadas para fabricar un uso experimental con el fin de comprender las consecuencias de estas y también proporcionar la probabilidad de explotación de la amenaza. La explotación le da a un intruso la capacidad de acceder a un sistema, un conjunto de aplicaciones, información sensible, etc. (Singh et al., 2024).

Informe de Resultados. Resultados resumidos de la prueba de penetración con problemas identificados y hallazgos detallados, incluidas vulnerabilidades y debilidades, impacto y riesgo, recomendaciones de remediación de componentes de software y dispositivos (Koman & Janiszewski, 2025).

Análisis de Impacto. En este paso se analizan los impactos que las vulnerabilidades descubiertas tienen en la organización y se determina una Calificación de Riesgo (República de Colombia, 2009).

Limpieza y Verificación. Se borra toda evidencia de la prueba de penetración de los sistemas y aplicaciones probados. Se verifica que todos los sistemas y aplicaciones hayan regresado a su estado original (pre-prueba) (Penetration Testing Execution Standard, 2023).

OSSTMM

Según ISECOM (2010), OSSTM es una metodología de pruebas de seguridad open source que permite ejecutar pruebas de penetración, análisis y medición de la seguridad operacional con el fin de robustecer la seguridad de la organización. Esta guía o manual se puede encontrar en el sitio oficial de isecom: https://isecom.org/research.html#content5-9d.

Fases de un Plan de Pruebas Según OSSTM (Open Source Security Testing Methodology Manual)

El Manual de la Metodología Abierta de Pruebas de Seguridad (OSSTMM), desarrollado por ISECOM, aborda la auditoría de seguridad desde una perspectiva estrictamente operacional y cuantitativa. Para garantizar la veracidad y la repetibilidad de los resultados, este estándar define siete pasos secuenciales diseñados para estructurar y ejecutar un plan de pruebas integral. El rigor de este flujo metodológico radica en su capacidad para evaluar la resiliencia de los activos a través de diferentes canales de interacción, permitiendo al auditor correlacionar las métricas de control operativo con las vulnerabilidades técnicas descubiertas. De este modo, la ejecución ordenada de las etapas no solo optimiza la fase de reconocimiento técnico, sino que fundamenta la toma de decisiones basada en hechos medibles y verificables. A continuación, se desglosa la naturaleza y el alcance de cada uno de estos pasos.

Preparación. En esta fase se determina el propósito de la prueba, el alcance, se eligen los componentes de software a probar. Identificar los posibles riesgos, determinar el presupuesto y el personal que llevará adelante la prueba también son definidos en esta fase (ISECOM, 2021).

Recopilación de Información. Se levanta toda la información relevante para poder llevar a cabo de manera exitosa la prueba. Para aplicaciones y servicios web es fundamental incluir sistemas, aplicaciones, servicios, direcciones IP, puertos, protocolos de comunicación, nombres de dominio, topologías de red, políticas de seguridad, entre otros (OWASP Foundation, 2023).

Análisis de Amenazas. Se identifican las posibles amenazas que podrían existir en los componentes de software y sistemas, por último, se evalúa el riesgo asociado a cada amenaza con el objetivo de categorizarlos (ISECOM, 2021).

Evaluación de Vulnerabilidades. En esta fase se identifican y evalúan las posibles vulnerabilidades existentes en los componentes que se van a probar. Esto se puede lograr gracias al uso de herramientas de análisis y exploración manual (Pooniya & Kumar, 2019; Koman & Janiszewski, 2025).

Análisis de Impacto. Se evalúa el impacto que las vulnerabilidades identificadas podrían generar en la organización y se determina el nivel de riesgo asociado.

Pruebas de Seguridad. En esta etapa se llevan a cabo las pruebas de seguridad utilizando diferentes técnicas de ataque, como el análisis de tráfico, la ingeniería inversa, el fuzzing, la inyección de código, entre otros (Singh et al., 2024). Con estas pruebas se pretende explotar las vulnerabilidades que existan en el sistema.

Informe de Resultados. El equipo encargado de ejecutar la prueba debe generar un informe tanto ejecutivo como operacional de las vulnerabilidades detectadas, su criticidad, impacto y forma en la que fue explotada (Koman & Janiszewski, 2025).

Verificación. En esta última etapa, se verifica que se hayan corregido las vulnerabilidades y debilidades identificadas durante las pruebas de seguridad. Esta verificación se realiza mediante la ejecución de un "re-test" (ISECOM, 2021).

ISSAF

La Information Systems Security Assessment Framework (ISSAF) es una metodología que proporciona una guía detallada para la realización de pruebas de seguridad en los sistemas de información. La guía se puede descargar desde el siguiente enlace https://sourceforge.net/projects/isstf/.

Fases de un Plan de Pruebas Según ISSAF (Information Systems Security Assessment Framework)

El Marco de Evaluación de Seguridad para Sistemas de Información (ISSAF), estructurado por el Open Information Systems Security Group (OISSG), se destaca en el ámbito de la ciberseguridad por su enfoque marcadamente operativo y procedimental. La idoneidad de este estándar radica en que no se limita al escaneo técnico de debilidades, sino que provee una guía exhaustiva para gobernar todo el ciclo de una evaluación de seguridad, abarcando desde las fases previas de planificación y gestión del alcance con las partes interesadas, hasta las etapas críticas de intrusión y post-evaluación. De acuerdo con el OISSG (s. f.), la ejecución de este marco exige el cumplimiento riguroso de etapas interconectadas, concebidas específicamente para desglosar la complejidad de los entornos de red y las aplicaciones web modernas bajo un criterio de auditoría formal. A continuación, se describen y analizan las fases fundamentales que constituyen este robusto marco de evaluación.

Planeación Preparación. En esta fase se identifican los canales de comunicación entre el dueño de la aplicación y el equipo encargado de ejecutar las pruebas, se establece el alcance de la prueba y la metodología a utilizar y por último se especifican los casos de prueba a ejecutar y los niveles para escalar los temas relacionados con las pruebas.

Evaluación. En esta fase se ejecuta el plan de pruebas, se detectan las vulnerabilidades en las aplicaciones objetivo.

Reporte. En esta fase se documentan y socializan las vulnerabilidades encontradas y las recomendaciones para la mitigación de estas vulnerabilidades.

Limpieza y Destrucción de Artefactos. Esta fase es muy simple y se enfoca en eliminar cualquier artefacto que haya quedado del proceso de ejecución de la prueba. Se da libertad al analista o profesional en seguridad que ejecutó la prueba para elegir cómo cifrar, desinfectar y destruir los datos creados durante el análisis.

Cada una de estas metodologías, marcos y/o estándares proporcionados en las secciones anteriores proponen una lista de pasos para probar la seguridad de los componentes de software y el entorno del software. Esto permitirá a los especialistas en seguridad formar una práctica de pruebas DAST justa y robusta para fortalecer sus aplicaciones y servicios web antes de que lleguen a producción (OWASP Foundation, 2023; Penetration Testing Execution Standard, 2023; ISECOM, 2021).

A partir del análisis comparativo realizado, se propone adoptar OWASP como el marco para los recursos de prueba DAST y aquí está el por qué:

La metodología OWASP es robusta, ya que ha crecido durante un poco más de 20 años (la primera versión se lanzó en 2001), se basa en las mejores prácticas de la industria y

estándares de seguridad (OWASP Foundation, 2023). Adicional a esto, el material de OWASP no es exactamente estático; hay un gran número de expertos contribuyendo y manteniéndolo.

Otro punto fuerte del enfoque OWASP es que se centra en encontrar los tipos de problemas que podrían estar asociados con las aplicaciones web. El enfoque OWASP no solo busca vulnerabilidades conocidas, sino que también comparte métodos y herramientas que se pueden usar para verificar amenazas específicas de aplicaciones web en cada una de las aplicaciones evaluadas, incluso proporcionando ejemplos de áreas problemáticas de código fuente en múltiples lenguajes de programación para que los desarrolladores puedan eliminar rápidamente la amenaza (OWASP Foundation, 2023), entre estos lenguajes de programación podemos encontrar Java, C# y PHP.

La metodología OWASP también proporciona instrucciones detalladas sobre cómo llevar a cabo las pruebas DAST, y específicamente cómo interpretar y analizar los resultados de las pruebas. Esto hace que sea más fácil para el equipo de seguridad y los desarrolladores trabajar juntos para remediar cualquier vulnerabilidad encontrada (Koman & Janiszewski, 2025; Somi, 2024).

Tácticas y Técnicas Utilizadas en la Evaluación de Aplicaciones y Servicios Web Mediante DAST

Según Somi (2024), la evaluación de aplicaciones y servicios web haciendo uso de DAST es efectiva gracias al uso de técnicas y tácticas que permiten identificar, catalogar, evaluar y explotar vulnerabilidades de manera adecuada, precisa y oportuna. Estas tácticas y técnicas se apalancan a su vez en herramientas que permiten automatizar tareas de manera muy práctica e incluso hacer uso de bases de datos públicas que contienen información global sobre las amenazas más conocidas. A continuación, se describen esas tácticas y técnicas de manera detallada.

Tácticas

Identificar el Flujo de Datos

Esta táctica tiene como objetivo entender cómo y qué datos fluyen en la aplicación (OWASP Foundation, 2023), por ejemplo, los datos que ingresa un usuario o que son enviados en una petición HTTP cómo son transportados, procesados y almacenados. Al comprender este flujo de datos es posible identificar y detectar algunas vulnerabilidades.

Para identificar este flujo de datos en una aplicación o servicio web, se pueden utilizar diversas técnicas, entre ellas:

- Escaneo de la aplicación
- Análisis de logs
- Auditoría de configuración
- Pruebas de penetración

Usualmente identificar el flujo de datos es una tarea compleja y requiere una alta comprensión del funcionamiento de la aplicación y los procesos de negocio asociados (Shostack,

2014). Además, se debe realizar en conjunto con otras tácticas de seguridad para asegurar una evaluación completa y precisa del componente evaluado, por tal razón, en muchas ocasiones se solicitan credenciales de un usuario con el cual se pueda acceder a la aplicación y tratar de entender el funcionamiento al navegar entre formularios o pantallas. En el caso de los servicios web, normalmente se solicitan cuerpos de mensajes o peticiones de ejemplo que puedan utilizarse.

Identificación de Amenazas

Usando esta táctica, el analista responsable de ejecutar la prueba se enfoca en identificar las posibles amenazas que enfrenta una aplicación o servicio web y cómo podrían ser explotadas por un atacante a partir de versiones o distribuciones de sistemas operativos, servicios o plataformas sobre las cuales se encuentra desplegada la aplicación, paquetes que se usaron en su construcción, entre otros (OWASP Foundation, 2023; PTES Technical Committee, s. f.). Para identificar estas amenazas, se pueden utilizar estas diversas técnicas:

- Escaneo de la aplicación
- Análisis de logs
- Pruebas de penetración
- Análisis de riesgos

Frameworks como OWASP brindan pautas para poder evaluar de manera completa y efectiva la aplicación e identificar las posibles amenazas en el componente destino del análisis. Es de aclarar que siempre se deben tomar utilizar varias técnicas o herramientas en este análisis para no caer en falsos positivos.

Identificación de Vulnerabilidades

La táctica de identificar vulnerabilidades tiene como objetivo descubrir XSS, SQLI y otros en una aplicación o servicio web y presentar soluciones para eliminar dichas detecciones (OWASP Foundation, 2021). Las vulnerabilidades podrían existir en capas de software como la capa de red, la capa de aplicación y la capa de base de datos.

También hay una variedad de herramientas, procesos y estrategias, incluyendo pruebas de penetración, análisis de registros, análisis de riesgos y escaneo de aplicaciones, que se pueden usar para identificar vulnerabilidades en una aplicación o servicio basado en la web.

Las pruebas de penetración son un método que se utiliza para simular ataques en la aplicación de servicios web para encontrar posibles vulnerabilidades de seguridad. En estas pruebas podemos utilizar técnicas como el escaneo de puertos y la ejecución de scripts predefinidos.

El análisis de logs es una metodología que busca sistemáticamente exposiciones y riesgos examinando los archivos de registro de una aplicación o servicio web. En algunas ocasiones los desarrolladores registran en los logs información sensible que permiten a un atacante obtener información relevante para explotar vulnerabilidades, por ejemplo, la cadena de conexión a una base de datos o información relevante sobre el motor usado.

Y por último, pero no menos importante, el escaneo de aplicaciones, con herramientas como Burp Suite, ZAP, NetSparker, etc. (Somi, 2024), ofrece una mirada exhaustiva a la aplicación o servicio web buscando esas molestas vulnerabilidades en diferentes capas.

Identificación de Exploits

La Detección de exploits busca identificar posibles explotaciones presentes en una aplicación o servicio y proponer formas de solucionarlas.

Para exponer el exploit, confiamos en el método de Identificación: buscando puertas, patrones y cosas extrañas en los WS de los solicitantes de recursos que puedan servir como objetivos para un atacante. Estas son las "puertas" que los malos usarán para alcanzar debilidades en la aplicación/servicio web.

Para reconocer probables explotaciones en vulnerabilidades de una aplicación o servicio web, podemos usar varias herramientas de Meterpreter como Burp Suite, ZAP, NetSparker, etc., para realizar pruebas exhaustivas y buscar o reconocer vulnerabilidades probables en numerosas capas como la capa de aplicación, capas de red, acceso a datos, entre otras (PTES Technical Committee, s. f.).

Además de algunos de estos métodos, otros también pueden usarse para lograrlo, como: pruebas de penetración, análisis de registros o incluso análisis de riesgo cualitativo-cuantitativo, permitiendo encontrar y solucionar posibles puntos de penetración de aplicaciones o servicios web.

Cabe señalar que para evaluar de manera exhaustiva y efectiva los posibles exploits de una aplicación o un servicio web de aplicación, es necesario utilizar una combinación de técnicas capaces de atacar varios puntos en la aplicación y la combinación de herramientas y técnicas mencionadas anteriormente para llevar a cabo una evaluación de seguridad adecuada y precisa de la aplicación o servicio.

Identificación y Evaluación de Impacto

La evaluación de impacto implica una evaluación de las vulnerabilidades identificadas en el paso anterior: identificación de vulnerabilidades. Las vulnerabilidades deben evaluarse en términos de cuán dañinas serían si se explotaran (por ejemplo, crítico, alto, medio, bajo), con igual consideración de cuán probable es que sean explotadas (FIRST, 2023; NIST, 2018).

También debemos tener en cuenta que la disponibilidad de la aplicación o servicio web, la reputación de la organización, la confidencialidad e integridad de los datos deben considerarse. Al comenzar a abordar una vulnerabilidad, ayuda tener una idea clara de la posible exposición.

Las políticas que aborden la vulnerabilidad señalada durante la evaluación de impacto se prescribirán posteriormente. Esto podría corregir, sellar las fallas y fortalecer la aplicación o servicio web para un escenario en el que la explotación no sería tan fácil, o podría llegar al punto de retirar el servicio/aplicación en cuestión mientras se espera que alguien entre y solucione el problema.

Técnicas

Escaneo de Puertos y Seguridad

Un escaneo de puertos es una técnica mediante la cual un atacante puede identificar los puertos abiertos que un dispositivo en particular puede presentar. De esta forma, se pueden relacionar posibles servicios con los puertos y encontrar deficiencias en el sistema (Lyon, 2009).

Una de las herramientas más usadas para realizar un escaneo de puertos es Nmap, esta herramienta es gratuita, muy utilizada por administradores de red y posterior a su instalación, ya sea en una distribución Linux o en Windows solo es necesario especificar la IP o el segmento de red para empezar con el análisis.

Cuando se inicia el escaneo, Nmap o un escáner de puertos envía un paquete a cada puerto de la dirección IP dada y espera para ver si el sistema remoto responde al paquete. Si se recibe una respuesta, el escáner de puertos reconoce que el puerto está abierto y registra la respuesta. Dependiendo de las opciones que ofrezca la herramienta y la naturaleza del escaneo, este proceso puede tardar desde unos minutos hasta unas horas.

El escaneo de puertos es un método para simplemente encontrar si un servicio existe en un sistema y hacerlo sin el permiso adecuado del propietario del sistema significaría acceso no autorizado o fraude (Ley 1273 de 2009). En el entorno empresarial, debe dejarse a personal autorizado y profesionales de escaneo de seguridad.

Fingerprinting

El fingerprinting es una forma increíble de descubrir vulnerabilidades potenciales y realizar pruebas de penetración. Con esta técnica se puede determinar el sistema operativo que se está ejecutando en el dispositivo objetivo del escaneo (ISECOM, 2010).

Esta técnica se lleva a cabo utilizando una variedad de herramientas y métodos, incluidos escaneos de puertos, examen de protocolos en red y escaneos de detección de servicios. Estos instrumentos y métodos pueden recopilar información sobre el sistema, como el sistema operativo en uso, los tipos de servicios de red disponibles, el sistema y tipo de software/firmware, y otros detalles similares.

El fingerprinting es una de las funciones más comunes, y para eso Nmap es una herramienta comúnmente utilizada, ya que tiene muchas características adicionales y es fácil de usar. Otras herramientas realizan reconocimiento de la misma o similar manera: Hping, Netscan Tools y Netcraft están entre ellas.

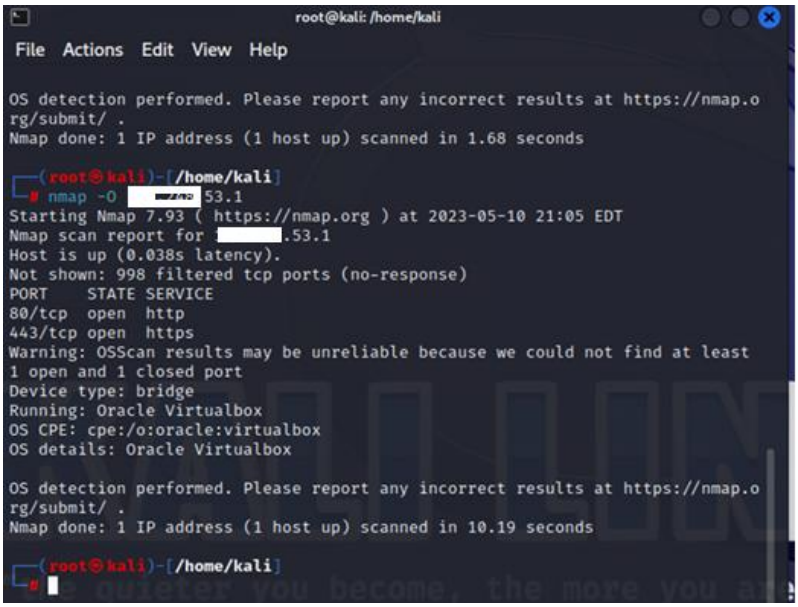
Igual que la técnicva de escaneo de puertos, el fingerprinting debe realizarse con el permiso y consentimiento del propietario del sistema y que no debe usarse para fines ilegales o no autorizados (Ley 1273 de 2009).

En la siguiente imagen se evidencia la ejecución del software Nmap en una distribución Kali Linux, con esta ejecución hacemos un escaneo de puertos a un servidor. En la Figura 5 se

evidencia que los puertos 80 y 443 se encuentran abiertos. Esto demuestra que es un servidor de aplicaciones o servicios web.

Figura 5

Ejecución de Nmap



Nota. La figura muestra la ejecución de la herramienta para hacer el escaneo de puertos.

Escaneo de Vulnerabilidades

El escaneo de vulnerabilidades es una técnica de seguridad informática que busca detectar las vulnerabilidades presentes en un sistema o aplicación (NIST, 2008; ISECOM, 2010). Esta técnica es muy importante para evitar que los atacantes exploten las debilidades del sistema y realicen acciones malintencionadas (NIST, 2008).

Para realizar un escaneo de vulnerabilidades, se utilizan diferentes herramientas y técnicas, como el escaneo de puertos, análisis de protocolos de red, pruebas de detección de servicios, análisis de código fuente, entre otros (Lyon, 2009; NIST, 2008). Gracias a estas

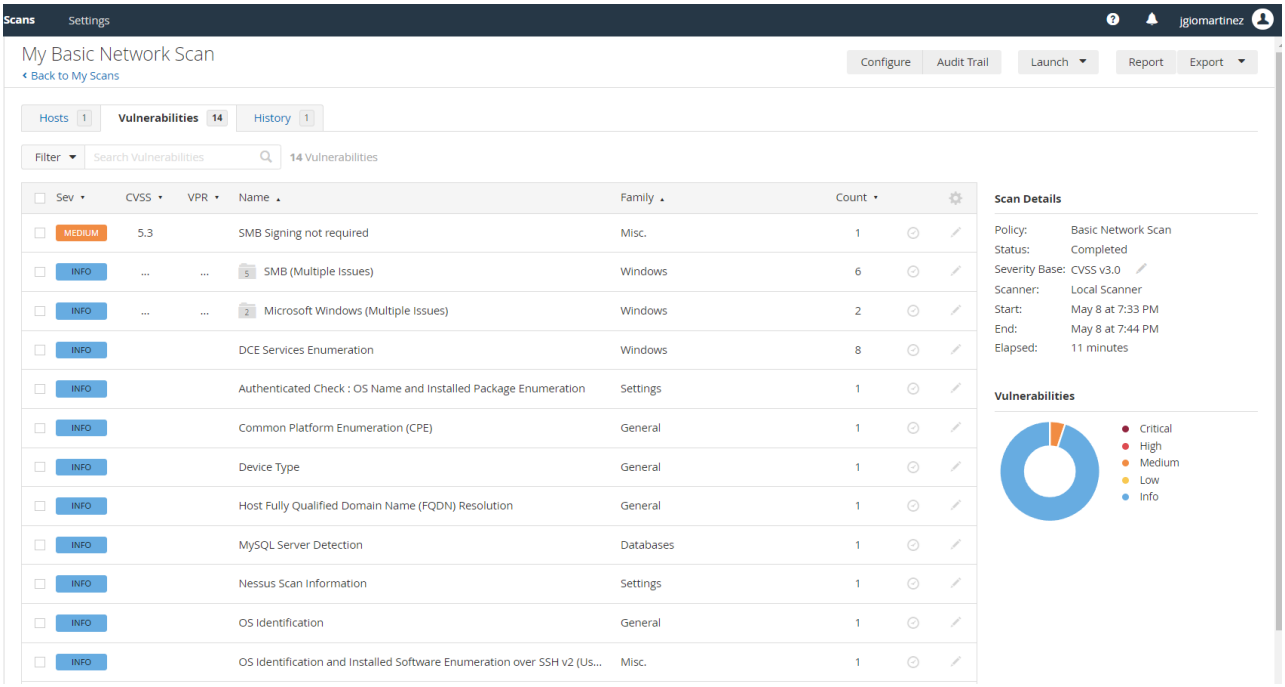
herramientas se pueden identificar y evaluar los diferentes vectores de ataque que existen y que son utilizados para explotar las vulnerabilidades en sistemas informáticos (OWASP Foundation, 2023). En nuestro caso, se omite la técnica de revisión de código fuente debido a que DAST es una prueba de caja negra y en este caso no tendríamos acceso al código fuente (Somi, 2024).

Nessus es una de las herramientas más utilizadas para el escaneo de vulnerabilidades, esta herramienta proporciona una amplia gama de opciones y una interfaz amigable para el usuario, Nessus puede instalarse en sistemas operativos windows y basados en linux. También hay otras herramientas como OpenVAS, Qualys, Rapid7 que se pueden utilizar para realizar pruebas de vulnerabilidad (Somi, 2024).

En la siguiente figura podemos ver un escaneo de vulnerabilidades realizado con la herramienta Nessus. Se puede evidenciar el listado de las vulnerabilidades encontradas en el host analizado.

Figura 6

Análisis con Nessus



Nota. Uso de la herramienta Nessus para hacer un escaneo de vulnerabilidades.

Nessus genera un reporte con las vulnerabilidades, su criticidad y un detalle con enlaces y posibles soluciones a la vulnerabilidad (Somi, 2024; Tenable, 2023).

Cuando se realiza una evaluación de seguridad en un sistema o aplicación, se pueden encontrar diversas vulnerabilidades que pueden ser de diferente gravedad o criticidad (FIRST, 2019; NIST, 2008). En este caso, se identifica una vulnerabilidad de criticidad media para la cual se proporciona información detallada.

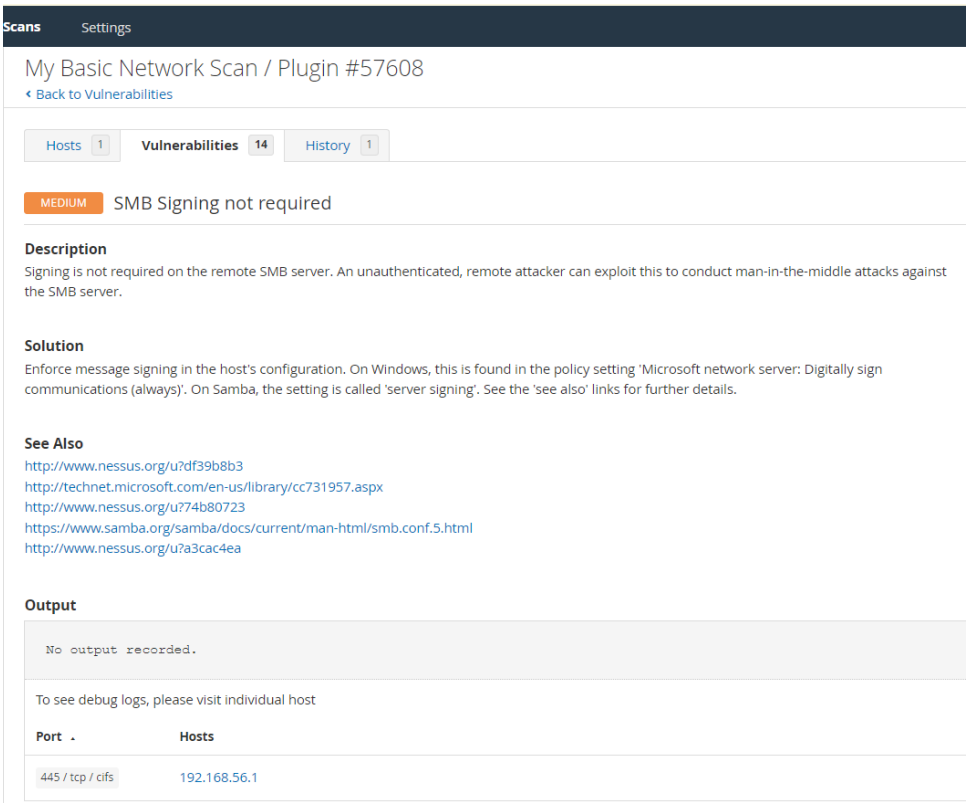
Es común que para cada vulnerabilidad encontrada se proporcione información como su descripción, la solución recomendada y enlaces donde se puede obtener más información (FIRST, 2019; OWASP Foundation, 2023). Esta información es importante tanto para los

profesionales encargados de la seguridad de la aplicación como para los desarrolladores, ya que les permite entender la naturaleza de la vulnerabilidad y los pasos que deben seguir para corregirla (Thool & Brown, 2025).

La siguiente imagen proporciona un ejemplo de cómo se presenta la información respecto a una vulnerabilidad de seguridad específica, y cómo se pueden proporcionar detalles sobre la misma. Esto es importante para abordar y corregir los problemas de seguridad de forma oportuna y eficiente.

Figura 7

Detalle de Vulnerabilidad Detectada con Nessus



Nota. Detalle de una vulnerabilidad detectada por Nessus y una recomendación para solucionarla.

Auditoría de Configuración

La auditoría de configuración es un proceso de revisión de los ajustes de configuración de un sistema o aplicación, con el objetivo de garantizar que estén en línea con las políticas de seguridad de la organización o esta configuración expone información sensible que puede ser utilizada por un atacante para explotar vulnerabilidades (ISECOM, 2010; NIST, 2008).

Para realizar una auditoría de configuración, se utilizan diferentes herramientas y técnicas que permiten recopilar e identificar información sobre los ajustes de configuración y su cumplimiento con los controles de seguridad establecidos. Se pueden usar herramientas de escaneo y auditoría para identificar brechas en la configuración y análisis de registro que proporcionan información detallada sobre los ajustes de configuración sin autorización y actividades sospechosas (ISECOM, 2010).

Entre las herramientas más utilizadas se destacan CIS-CAT, Microsoft Baseline Security Analyzer, OpenVAS, Nessus. Además, algunas herramientas para la gestión de configuraciones como Chef, Puppet y Ansible incluyen funcionalidades para la auditoria de la configuración (Pooniya & Kumar, 2019).

Pruebas de penetración

La prueba de penetración es la práctica de simular un ataque real a un sistema o una aplicación para evaluar la seguridad de este. El propósito de la prueba de penetración es descubrir fallos e inseguridades del sistema, para que puedan ser corregidos y reforzar la seguridad del componente evaluado (PTES Technical Committee, s. f.; ISECOM, 2010).

En una prueba de penetración, los profesionales de seguridad hacen uso de diferentes herramientas, tácticas y procedimientos que finalmente les permiten examinar y verificar cada

parte del sistema. Es común dividir el proceso en tres fases: análisis de vulnerabilidades, explotación e informe.

Durante el análisis de vulnerabilidades, se examina el sistema en busca de debilidades y se evalúa su potencial impacto. En esta fase, se utilizan herramientas como nmap, Nessus, OpenVAS, entre otras, para el escaneo de puertos, identificación de servicios y aplicaciones, y la búsqueda de vulnerabilidades (PTES Technical Committee, s. f.).

En la fase de explotación, se intenta explotar las vulnerabilidades detectadas en la fase anterior para obtener acceso no autorizado o información confidencial. En muchas ocasiones se utilizan herramientas de prueba de penetración como Metasploit, CobaltStrike o Armitage para automatizar parte del trabajo y evitar errores humanos (PTES Technical Committee, s. f.).

En la fase de informe final se presentan 2 reportes, el primero orientado a la dirección en el que se muestra el porcentaje de vulnerabilidades agrupadas por criticidad y el segundo orientado al equipo de desarrollo encargado de gobernar la aplicación, entendiéndose como gobierno el analizar, diseñar, desarrollar y soportar la aplicación o componente (PTES Technical Committee, s. f.; OWASP Foundation, 2023).

Kali Linux es una distribución de Linux especialmente diseñada para la seguridad informática y el pentesting. Esta distribución incluye una gran cantidad de herramientas de software libre que pueden ser utilizadas para llevar a cabo diferentes tipos de pruebas de penetración en sistemas y redes, entre ellos los servicios y aplicaciones web (PTES Technical Committee, s. f.).

Entre las herramientas más populares incluidas en Kali Linux se encuentran Wireshark, que permite capturar y analizar el tráfico de red, NMap, como ya lo hemos indicado es una herramienta de escaneo de puertos, Metasploit, una plataforma para pruebas de penetración

automatizadas que permite la ejecución de scripts predefinidos y personalizados. Estas herramientas son comúnmente utilizadas por los profesionales de seguridad informática, no sobra aclarar que deben usarse dentro del marco ético y legal.

Análisis de Logs

Cada aplicación tiene registros/archivos donde se almacena información para que se puedan diagnosticar problemas o registrar cualquier actividad que ocurra dentro de la aplicación en tiempo de ejecución. En algunos casos, los desarrolladores no hacen el mejor trabajo de desarrollo seguro y ponen demasiada información en los registros que puede ser utilizada por un atacante para explotar aún más la vulnerabilidad. Encontramos cosas como cadenas de conexión a bases de datos, credenciales para sistemas externos, directorios, etc., dentro de esta información (OWASP Foundation, 2023).

Para realizar un análisis de logs se utilizan diferentes herramientas, técnicas y es necesario tener acceso a los registros del sistema o aplicación, que generalmente se almacenan en archivos o bases de datos.

Algunas herramientas que podrían ser utlizadas son Splunk, ELK Stack (Elasticsearch, Logstash, Kibana) o Graylog. Aquí hay algunas utilidades disponibles que hacen que este proceso sea bastante simple para recopilar/analizar/visualizar registros (Thool & Brown, 2025).

El análisis de registros puede realizarse manual o automáticamente, dependiendo de los registros y del uso previsto. Lo que normalmente hacemos es aplicar algunas técnicas para localizar información útil del registro, por ejemplo, minería de datos, correlación de eventos y análisis de patrones (ISECOM, 2010).

Las medidas de seguridad, incluyendo la política de gestión de registros, el cifrado de la información de los registros, la restricción de acceso a la información por parte del personal

autorizado, se utilizan para prevenir riesgos de la información de los registros. En consecuencia, el método implementado por computadora puede mitigar el riesgo de uso no autorizado de los datos registrados (NIST, 2008; OWASP Foundation, 2023).

Análisis de Vulnerabilidades Ejecutando Pruebas DAST

En esta sección se presenta el proceso necesario para identificar vulnerabilidades en aplicaciones web utilizando el enfoque DAST. Para realizar esta tarea se utilizaron las herramientas OWASP ZAP y OWASP Juice Shop, la primera es una de las herramientas DAST más reconocidas de código abierto y la segunda es una aplicación web vulnerable de manera intencionada y muy usada en entornos educativos. Las dos herramientas son proyectos oficiales de OWASP.

Identificación del Objetivo del Análisis

Para fines de esta monografía el objetivo fue la aplicación web OWASP Juice Shop, una aplicación moderna de comercio electrónico desarrollada en Node.js, Express y Angular. Esta aplicación simula un entorno real donde los usuarios pueden buscar, seleccionar, ver el detalle, agregar al carrito y comprar productos.

Las vulnerabilidades detectadas mediante el enfoque DAST serán también evaluadas de manera manual con el objetivo de determinar la efectividad de DAST y observar el porcentaje de falsos positivos reportados por este análisis.

Descripción del Entorno de Prueba

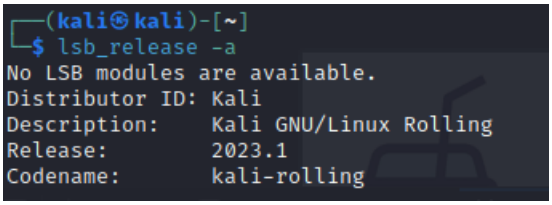
Entorno y Arquitectura

La prueba se ejecutó en una máquina virtual Kali-Linux, las dos herramientas se ejecutaron en contenedores Docker. La aplicación web OWASP Juice Shop estaba siendo ejecutada por el puerto http 3000. La versión de OWASP Juice Shop fue la 19.0.0 y la versión de ZAP fue la 2.16.1.

En las siguientes figuras se muestra la distribución de Linux usada, las imágenes de Docker y el contenedor de Juice Shop en ejecución.

Figura 8

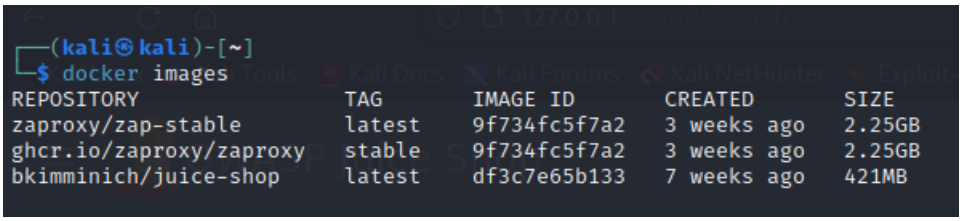
Distribución de Linux Utilizada para la Ejecución del Análisis



Nota. Comando ejecutado para obtener la distribución actual de Linux.

Figura 9

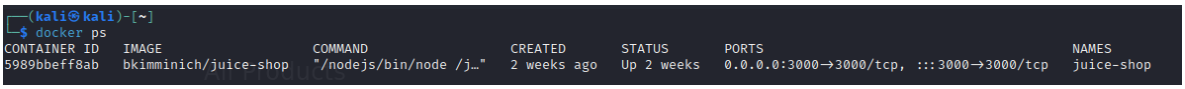
Imágenes Utilizadas en Docker para la Ejecución del Análisis



Nota. Comando ejecutado para visualizar las imágenes utilizadas en Docker para la ejecución del análisis.

Figura 10

Contenedor de Juice-Shop Escuchando por el Puerto 3000



Nota. Se muestra el contenedor de Juice-Shop utilizado para la prueba.

Configuración de la Prueba

El escaneo fue generado utilizando el siguiente comando:

$$zap - baseline.py \ -t \ http://127.0.0.1:3000 \ -r \ /zap/wrk/juiceshop - baseline.html$$
$$-j \ /zap/wrk/juiceshop - baseline.json$$

Con este comando se ejecutó un escaneo automatizado, el destino de la prueba fue el sitio http://127.0.0.1:3000, en esta ruta se encontraba desplegada la aplicación web OWASP Juice Shop, en el parámetro -r se indica la ruta del reporte HTML y en el parámetro -j se indica la ruta del reporte en formato json.

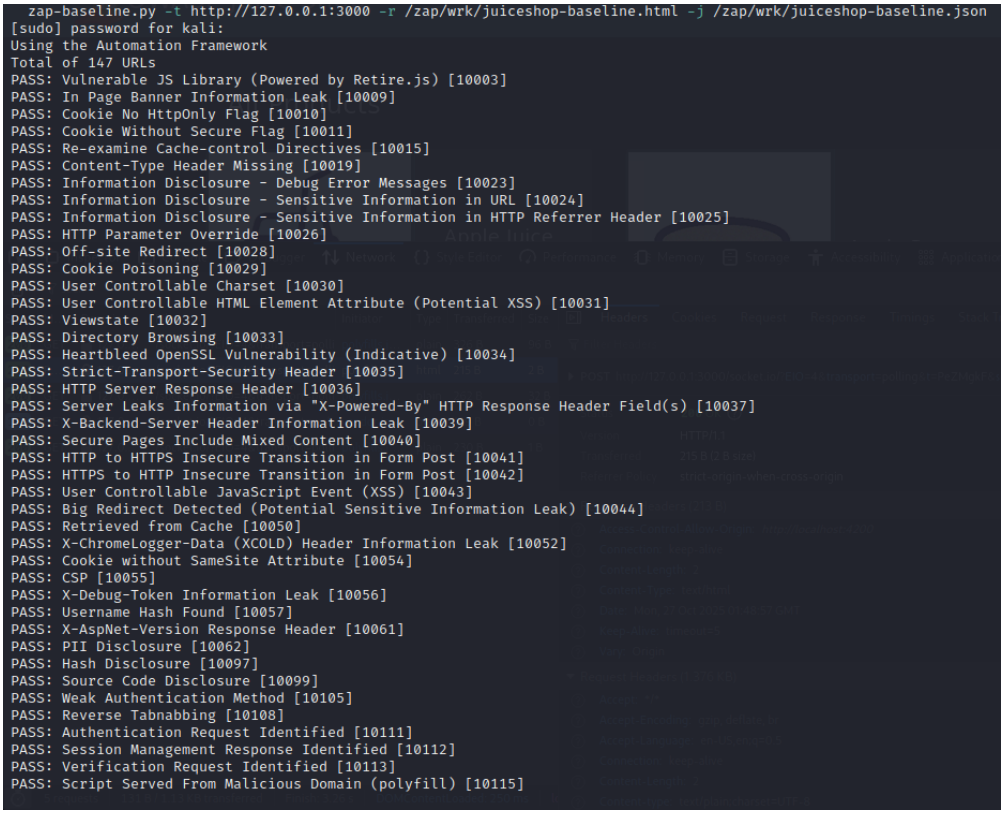
Ejecución del Análisis

Ejecución del Comando

Al ejecutar el comando se puede evidenciar cada una de las vulnerabilidades que el script empieza a evaluar, debido a que la aplicación es vulnerable por naturaleza son varias las que el script detecta de manera rápida, (**ver Figura 11**).

Figura 11

Análisis en Ejecución



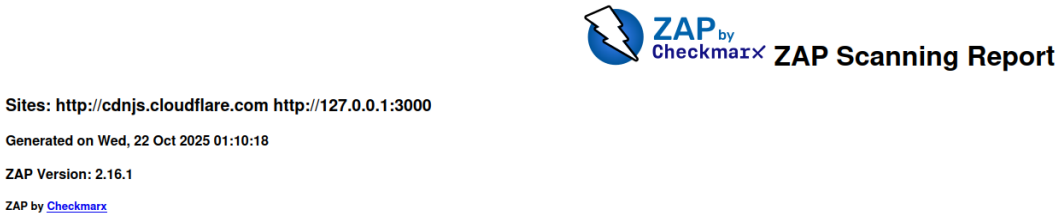
Nota. Análisis de vulnerabilidades en ejecución.

Tiempo de Ejecución del Análisis

El tiempo de duración del análisis fue de 15 minutos, iniciando el 21 Oct 2025 a las 19:55 y finalizando a las 20:10 GMT -5 (**ver Figura 12**):

Figura 12

Fecha y Hora del Análisis



Nota. Evidencia que muestra la fecha y hora de finalización del análisis.

Resultado del Análisis

La distribución de las vulnerabilidades detectadas, categorizadas según su impacto y nivel de criticidad, se presenta en la **Figura 13**. Esta métrica permite priorizar los vectores de ataque que requieren una remediación inmediata.

Figura 13

Resumen de Alertas

Risk Level	Number of Alerts
High	0
Medium	4
Low	7
Informational	4
False Positives:	0

Nota. Alertas detectadas y categorizadas por nivel de riesgo.

Es de aclarar que, aunque ZAP indica que no existen falsos positivos, esto no garantiza que el análisis esté completamente libre de errores, solo indica que ninguna vulnerabilidad fue

catalogada explícitamente como falsa en el reporte. Para garantizar la fiabilidad de los resultados, se realizó una revisión manual de las alertas reportadas, de esta forma, fue posible validar cuáles vulnerabilidades eran reproducibles y cuáles no. En este caso todas las vulnerabilidades reportadas fueron reproducidas mediante el análisis manual y se garantizó que no existen falsos positivos. Esta es una de las ventajas de estos tipos de análisis, cada vez las herramientas existentes que aplican DAST son más fiables y generan un número muy pequeño de falsos positivos garantizando su confiabilidad e incrementando la aplicabilidad en herramientas web.

Alertas Detectadas

Tras el análisis de resultados, se obtuvo una cuantificación de las alertas de seguridad según su criticidad y recurrencia (**ver Figura 14**).

Figura 14

Alertas Detectadas en el Análisis de Vulnerabilidades

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	11
Cross-Domain Misconfiguration	Medium	11
Missing Anti-clickjacking Header	Medium	8
Session ID in URL Rewrite	Medium	12
Cross-Domain JavaScript Source File Inclusion	Low	10
Dangerous JS Functions	Low	2
Deprecated Feature Policy Header Set	Low	11
Insufficient Site Isolation Against Spectre Vulnerability	Low	10
Private IP Disclosure	Low	1
Timestamp Disclosure - Unix	Low	16
X-Content-Type-Options Header Missing	Low	12
Information Disclosure - Suspicious Comments	Informational	3
Modern Web Application	Informational	11
Storable and Cacheable Content	Informational	1
Storable but Non-Cacheable Content	Informational	10

Nota. Enumeración y cardinalidad de las alertas detectadas por cada nivel de riesgo.

Cada vulnerabilidad detectada tiene un enlace al detalle de esta donde se explica la vulnerabilidad, las URL en las que se detectó y enlaces a la documentación. En la **Figura 15**, se muestra el detalle de la primera vulnerabilidad “Content Security Policy (CSP) Header Not Set”.

Figura 15

Detalle de Alerta

Alert Detail

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://127.0.0.1:3000
Method	GET
Parameter	
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:3000/
Method	GET
Parameter	
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:3000/ftp
Method	GET
Parameter	
Attack	

Nota. Detalle de una alerta detectada, su descripción y las URL en las que aparece.

Como evidencia técnica del análisis ejecutado sobre el entorno de pruebas, se procedió a exportar el informe consolidado desde la herramienta OWASP ZAP, este reporte se encuentra disponible en el apéndice C.

Verificación Manual de Vulnerabilidades

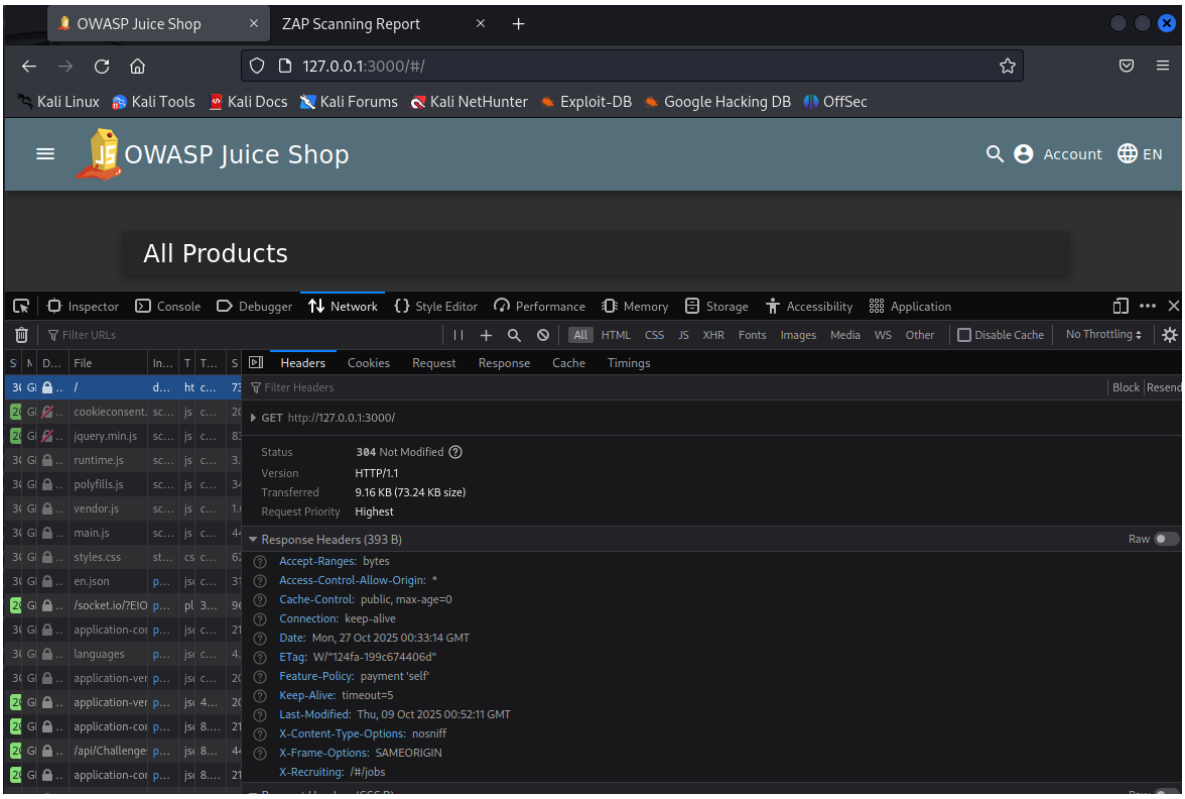
Para verificar que no existan falsos positivos se evalúan algunas vulnerabilidades de manera manual. A continuación, se muestran las vulnerabilidades que fueron evaluadas de esta forma.

Content Security Policy (CSP). Esta vulnerabilidad fue detectada por ZAP en la url http:127.0.0.1:3000, y como se puede visualizar en la Figura 16, efectivamente el header Content-Security-Policy no es enviado en el response del recurso http.

Para identificar la ausencia del header, simplemente accedemos al recurso utilizando un navegador web y examinamos los headers que llegan en el response de la petición http del recurso.

Figura 16

Análisis Manual de Vulnerabilidad de Tipo CSP

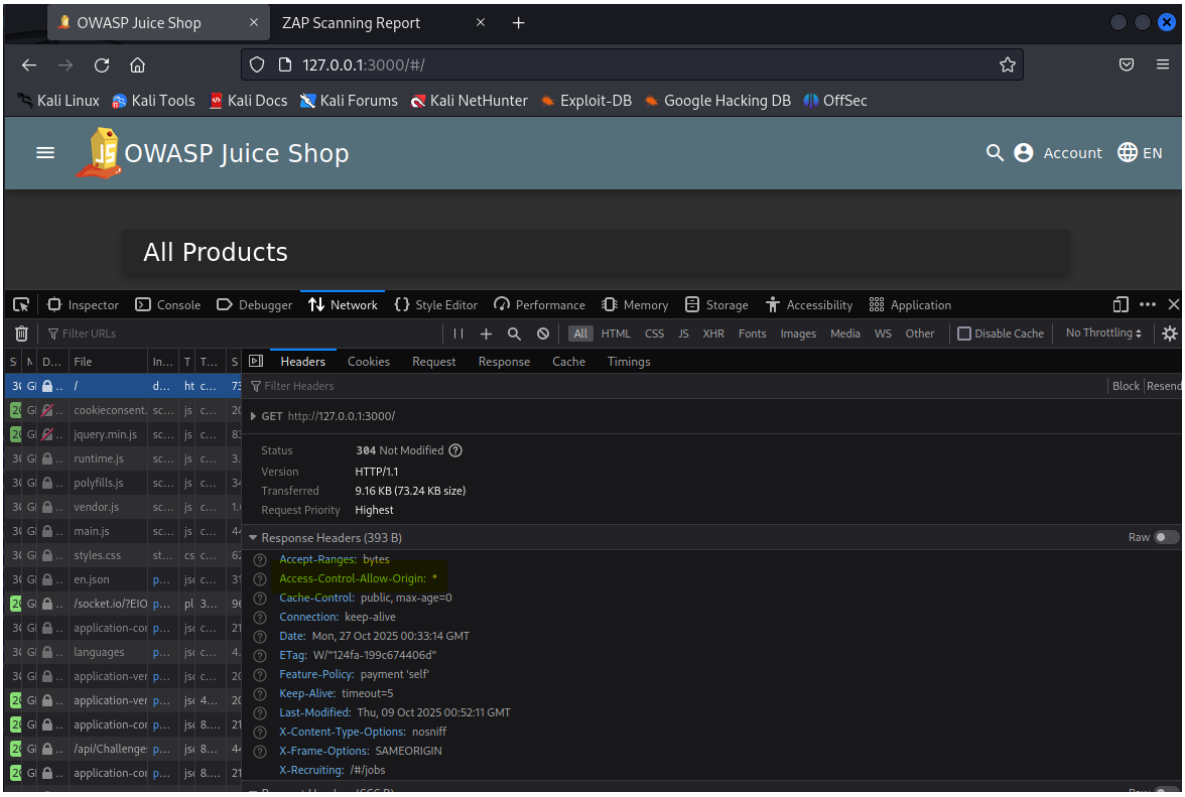


Nota. Evidencia de detección manual de la vulnerabilidad de tipo CSP.

Cross Domain Miss Configuration. Se evidencia esta vulnerabilidad al evaluar los headers del response de la petición HTTP hacia el recurso http:127.0.0.1:3000, en los headers se encuentra el “Access-control-allow-origin” con valor “*”. En la Figura 17 se puede ver lo expuesto en el párrafo anterior.

Figura 17

Cross Domain Miss Configuration

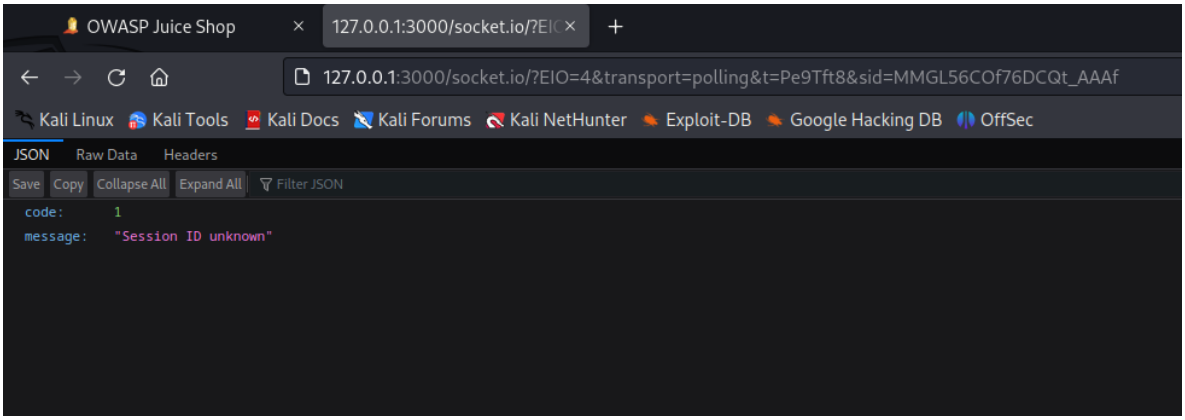


Nota. Evidencia de detección manual de la vulnerabilidad de tipo cross domain miss configuration.

Session ID in URL Rewrites. Esta vulnerabilidad se puede reproducir al evidenciar que en la URL existe un parámetro que se envía por querystring en las peticiones GET utilizando la URL http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=Pe9Tft8&sid=MMGL56COf76DCQt_AAAf. El ID de sesión es MMGL56COf76DCQt_AAAf, de esta forma se puede suplantar la identidad del usuario actual al robar una sesión y utilizarla simplemente agregando el session ID en la URL. En la Figura 18 se puede visualizar que en una petición GET se recibe el sid por parámetro.

Figura 18

Session ID en Petición GET

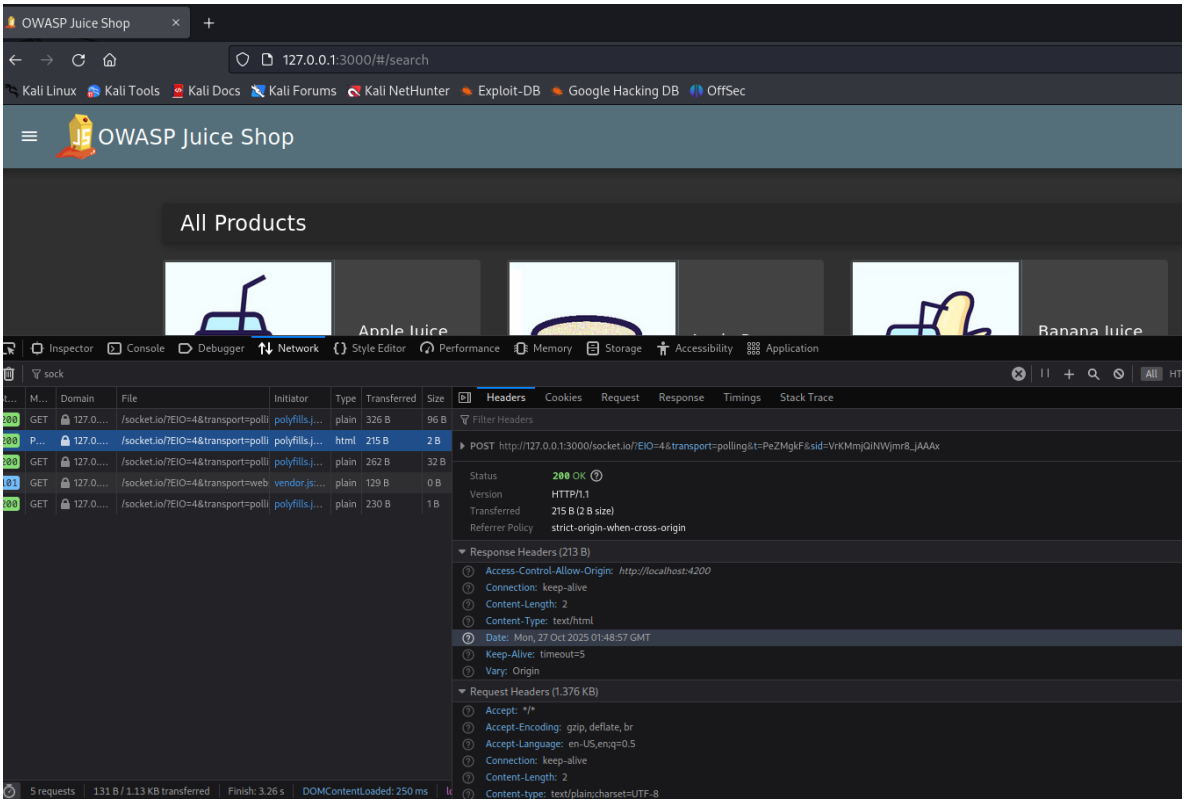


Nota. Evidencia de detección manual de la vulnerabilidad que permite inyectar el ID de sesión en la petición GET HTTP.

En la petición POST también se evidencia el envío del parámetro “sid” en la URL (**ver Figura 19**).

Figura 19

Session ID en Petición POST



Nota. Evidencia de detección manual de la vulnerabilidad que permite inyectar el ID de sesión en la petición POST HTTP.

Evaluación Efectiva de la Prueba

La prueba tuvo una duración de 15 minutos aproximadamente y en esta se detectaron las 15 vulnerabilidades que se enumeran en la Tabla 2:

Tabla 2

Vulnerabilidades Detectadas en Análisis DAST

	Vulnerabilidad
1.	Content Security Policy (Header Not Set)
2.	Cross-Domain Missconfiguration
3.	Missing Anti-clickjacking Header
4.	Session ID in URL Rewrite
5.	Cross-Domain JavaScript Source File Inclusion
6.	Dangerous JS Functions
7.	Deprecated Feature Policy Header Set
8.	Insufficient Site Isolation Against Spectre Vulnerability
9.	Private IP Disclosure
10.	Timestamp Disclosure – Unix
11.	X-Content-Type-Options Header Missing
12.	Information Disclosure – Suspicious Comments
13.	Modern Web Application
14.	Storable and Cachable Content
15.	Storable but Non-Cachable Content

Nota. Lista de vulnerabilidades detectadas en el análisis ejecutado.

Para cada vulnerabilidad detectada, el reporte generado por OWASP ZAP presenta un detalle con la descripción, la URL o las URL en las que se detectó el vector de exposición y enlaces a páginas de internet en las que se muestra como mitigar la vulnerabilidad. En la **Figuras 20 y 21** se ilustra a modo ejemplo, el desglose detallado de una vulnerabilidad específica y sus respectivas referencias de mitigación.

Figura 20

Referencias para Solucionar una Vulnerabilidad

Solution	Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html https://www.w3.org/TR/CSP/ https://w3c.github.io/webappsec-csp/ https://web.dev/articles/csp https://caniuse.com/#feat=contentsecuritypolicy https://content-security-policy.com/
CWE Id	693
WASC Id	15
Plugin Id	10038

Nota. Detalle de las referencias que se pueden utilizar para solucionar la vulnerabilidad detectada.

Figura 21

Detalle de la Vulnerabilidad

Alert Detail

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://127.0.0.1:3000
Method	GET
Parameter	
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:3000/
Method	GET
Parameter	
Attack	
Evidence	
Other Info	

Nota. Detalle de una vulnerabilidad detectada de tipo CSP.

Para garantizar que no existieran falsos positivos, las vulnerabilidades fueron evaluadas de manera manual, el resultado fue satisfactorio, todas las vulnerabilidades evaluadas y reportadas por OWASP ZAP fueron confirmadas.

Conclusión de la Prueba

El análisis realizado mediante la herramienta ZAP, en solo 15 minutos pudo detectar un total de 15 vulnerabilidades reales sin falsos positivos en una aplicación de tamaño considerable como lo es un e-commerce. El detalle de las vulnerabilidades fue muy bien documentado y de manera automática mediante reportes en formato JSON y HTML, esta documentación puede ayudar de manera considerable a los equipos de desarrollo de software en la evaluación y mitigación de estas vulnerabilidades.

La prueba DAST demuestra su efectividad en el análisis de vulnerabilidades al detectar, de manera rápida y confiable, los defectos de seguridad que puede tener una aplicación o servicio web. De esta forma, puede apoyar a los equipos de desarrollo de software y de ciberseguridad de una compañía a analizar los componentes web previo a su despliegue en producción. De esta forma, es muy fácil integrar el análisis de vulnerabilidades mediante DAST en el ciclo de desarrollo del software para garantizar el despliegue de aplicaciones o servicios web seguros en un entorno productivo.

Modelo Propuesto

A continuación, se propone un modelo de aplicación del proceso DAST orientado a una implementación organizacional.

Fases del Proceso

Esta sección describe las etapas clave para la implementación efectiva del análisis DAST, desde la identificación del objetivo hasta la socialización de resultados. El proceso incluye la planificación del escaneo, ejecución, análisis de hallazgos, mitigación y retroalimentación, asegurando una integración estructurada en el ciclo de desarrollo seguro.

Identificación del Objetivo

Esta etapa tiene pretende identificar la aplicación o componente web que se quiere vulnerar, sus entornos (testing, staging y producción), el servidor web y sistema operativo sobre el cual será desplegado. En esta etapa se deben reunir el equipo de desarrollo y el líder de seguridad para ejecutarla.

Planificación del Análisis

Luego de haber identificado el objetivo del análisis, el equipo de seguridad debe planificar el análisis definiendo el alcance, las herramientas que debe utilizar, las credenciales de acceso necesarias y el tiempo aproximado del análisis.

Ejecución del Análisis

Luego de haber recopilado toda la información necesaria y planificado el análisis, el equipo de seguridad debe proceder con la ejecución del mismo, para esto se debe apoyar en herramientas como OWASP ZAP, Burp Suite o similares. Estas herramientas se especializan en el análisis de vulnerabilidades sobre servicios y aplicaciones web.

Análisis de Resultados

Al finalizar el análisis, la herramienta utilizada generará un listado de hallazgos clasificados por tipo y severidad, estos hallazgos deben ser analizados por el equipo de seguridad para definir la criticidad de estos y de paso descartar falsos positivos.

Reporte y Entrega

El equipo de seguridad debe documentar las vulnerabilidades encontradas, adjuntar el reporte que la herramienta genera, este adjunto servirá como referencia al equipo de desarrollo para mitigar las vulnerabilidades según el lenguaje de desarrollo o servidor web que soporta la aplicación web. También se deberá generar una matriz de riesgo donde se clasifiquen las vulnerabilidades según su criticidad.

Mitigación y Verificación

En esta fase el equipo de desarrollo debe mitigar las vulnerabilidades reportadas en la fase anterior tomando como base las recomendaciones y la documentación proporcionada en la fase anterior. Posterior a la mitigación, el equipo de desarrollo debe notificar al equipo de seguridad para que este ejecute un retest y se confirme la mitigación.

Socialización

En esta etapa, el equipo de seguridad debe compartir los resultados a nivel técnico y ejecutivo con el equipo de desarrollo y líder del proyecto respectivamente. En los resultados se deben mostrar las vulnerabilidades detectadas en la primera ejecución y las existentes posterior a la mitigación. La matriz de riesgo debe evidenciar las vulnerabilidades mapeadas según su probabilidad de ocurrencia e impacto.

Formatos de Reporte de Resultados

En esta sección se proponen dos formatos de reporte para comunicar los hallazgos del análisis DAST: uno técnico, orientado a equipos de desarrollo y seguridad, con información detallada por vulnerabilidad; y otro ejecutivo, diseñado para niveles gerenciales, con indicadores resumidos y visualizaciones que facilitan la toma de decisiones estratégicas. Los modelos estructurados para ambos tipos de informe se presentan detalladamente en el Apéndice D.

Socialización en Diferentes Niveles Operativos

La socialización de los hallazgos del análisis DAST es una parte importante para mostrar las vulnerabilidades descubiertas a todas las partes interesadas en el flujo de desarrollo y operaciones de software. Esta sección establece un plan de comunicación adecuado para diferentes niveles de gestión organizacional/operativa, diseñado para ayudar en la toma de decisiones, la priorización de medidas correctivas y para relacionar las actividades del SGSI con el negocio.

El modelo define qué información brindar a quién (desarrollo, equipo de QA, líderes tecnológicos, PM, ejecutivos, etc.), a qué nivel (resumen, detalle...), qué formato utilizar (tableros, informes técnicos, resúmenes ejecutivos...) y con qué frecuencia, vinculado al nivel de criticidad de los hallazgos. Esto permite convertir los hallazgos del escaneo en actividades prácticas y cuantificables en el flujo de trabajo de DevSecOps.

A continuación, se elabora una tabla en la que se detalla información relevante para cada role existente en una empresa.

Tabla 3

Tabla de Socialización de Resultados

Nivel	¿Qué se comunica?	Formato	Frecuencia
Desarrolladores	Detalle técnico + rutas afectadas	Informe técnico	Inmediato
Equipo QA	Escenarios afectados + pruebas de regresión	Informe técnico	Por ciclo
Gestión/PMs	Severidad, impacto, tiempos de remediación	Informe ejecutivo	Semanal
Alta Dirección (CISO/CTO)	Tendencias, riesgos críticos, cumplimiento	Informe ejecutivo + KPIs	Mensual

Nota. Información que se debe socializar a cada rol existente en la empresa. Se incluye el formato a utilizar y la frecuencia en que se comunica.

Conclusiones

En cumplimiento del primer objetivo propuesto, se logró describir con rigor el proceso DAST (Dynamic Application Security Testing) en el marco de las metodologías de evaluación de seguridad en aplicaciones y servicios web, resaltando su relevancia como técnica de análisis dinámico que permite identificar vulnerabilidades desde una perspectiva externa al sistema. Se abordaron las principales tácticas y técnicas empleadas en este tipo de pruebas, tales como la inyección de entradas maliciosas, la detección de comportamientos anómalos y el análisis de respuestas del sistema ante distintos vectores de ataque. Asimismo, se identificaron herramientas especializadas que facilitan la estructuración de un plan de pruebas efectivo, entre ellas OWASP ZAP, Burp Suite y Acunetix (Invicti, 2024), cuya aplicación práctica contribuye significativamente al fortalecimiento de la postura de seguridad en entornos web.

Se ejecutó un análisis de vulnerabilidades haciendo uso de la herramienta OWASP ZAP al aplicativo OWASP Juice Shop. Este análisis duró solo 7 minutos dando como resultado la identificación de 67 amenazas reales. Esto demuestra la efectividad de un análisis DAST y la importancia de incluirlo en el ciclo de desarrollo del software previo a un despliegue en producción.

El modelo propuesto para realizar un análisis DAST sobre aplicaciones y servicios Web está compuesto por 7 fases entre las cuales se encuentran: Identificación del objetivo, planificación del análisis, ejecución del análisis, análisis de resultados, reporte y entrega, mitigación y verificación, y por último la socialización de los resultados. En esta última fase se propone el uso de 2 formatos, uno orientado a la dirección y presentado de manera ejecutiva y el otro orientado de manera operativa al equipo de desarrollo y líderes del proyecto.

Recomendaciones

Con base en los resultados obtenidos, es recomendado el uso de pruebas con el enfoque DAST en el ciclo de desarrollo del software y previo a un despliegue en producción con el objetivo de detectar vulnerabilidades que puedan ser explotadas ocasionando diferentes impactos a la empresa dueña del activo explotado, entre ellos reputacional y monetario. Un análisis DAST puede tardar pocos minutos, entre 10 y 60 minutos dependiendo del tamaño y cantidad de componentes a examinar. En pocos minutos puede identificar vulnerabilidades relevantes que el equipo de Desarrollo puede mitigar tomando como base la robusta documentación que se encuentra en sitios como OWASP.

Las compañías deben utilizar un enfoque DAST apalancado de frameworks como OWASP que ofrecen una amplia gama de recursos y herramientas que permiten a los profesionales de seguridad e incluso al equipo de desarrollo detectar fácilmente el top 10 de vulnerabilidades existentes en aplicaciones y servicios web.

Normalmente el equipo de seguridad informática se encuentra desacoplado de los equipos de desarrollo de software, esto hace que los procesos de análisis de vulnerabilidades se hagan un poco tediosos y traumáticos para el despliegue de servicios y aplicaciones. En la etapa 6 del modelo propuesto “Mitigación y verificación” se propone un trabajo en equipo para garantizar la rápida mitigación de las vulnerabilidades. Si en los equipos de desarrollo existiera un rol encargado de velar por la seguridad de sus aplicaciones desde la codificación, se podría fácilmente llegar a la fase 6 con la menor cantidad de vulnerabilidades posibles.

Referencias Bibliográficas

Arias, F. G. (2012). *El proyecto de investigación: Introducción a la metodología científica (6.ª ed.)*. Episteme. https://www.researchgate.net/publication/301894369_EL_PROYECTO_DE_INVESTIGACION_6a_EDICION

Bassos, F., & Pretto, J. (2019). *Automatizando la resolución de problemas en competencias de Seguridad Informática* [Tesina de grado, Universidad Nacional de La Plata]. Repositorio Institucional de la UNLP. http://sedici.unlp.edu.ar/handle/10915/119997

FIRST. (2019). *Common vulnerability scoring system (CVSS) v3.1 specification document*. Forum of Incident Response and Security Teams. https://www.first.org/cvss/specification-document

Francis, M. (2025). Historical evolution of security testing for web applications. *International Journal of Computer Applications, 186*(82). https://www.ijcaonline.org/archives/volume186/number82/ijca-submission-historical-evolution-of-security-testing-for-web-applications

Hernández-Sampieri, R. & Mendoza, C (2018). *Metodología de la investigación. Las rutas cuantitativa, cualitativa y mixta*. McGraw Hill Education.

Invicti. (2024). *Acunetix: Web Application Security Scanner* [Software de computación]. https://www.acunetix.com/

ISECOM. (2021). *Open source security testing methodology manual (OSSTMM)*. https://isecom.org/research.html

Koman, J., & Janiszewski, M. (2025). SCAnME – Scanner comparative analysis and metrics for evaluation. *International Journal of Information Security, 24*(3), Article 147. https://doi.org/10.1007/s10207-025-01054-8

López Álvarez, D. M. (2020). Método para el desarrollo de software seguro basado en la ingeniería de software y ciberseguridad. *INNOVA Research Journal, 5*(3.1), 263–280. https://doi.org/10.33890/innova.v5.n3.1.2020.1440

Lyon, G. (2009). *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure.com LLC. https://nmap.org/book/

NIST. (2008). *Guide to general server security (SP 800-123)*. National Institute of Standards and Technology. https://csrc.nist.gov/publications/detail/sp/800-123/final

Open Information Systems Security Group. (s.f.). *Information Systems Security Assessment Framework (ISSAF): Technical Guide*. http://www.oissg.org/issaf

OWASP Foundation. (2023). *OWASP web security testing guide*. https://owasp.org/www-project-web-security-testing-guide/

Penetration Testing Execution Standard. (2023). *PTES technical guidelines*. http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

Pooniya, R., & Kumar, V. (2019). A comparative study of automated web application vulnerability scanners. *International Journal of Advanced Computer Science and Applications, 10*(3), 392–398. https://doi.org/10.14569/IJACSA.2019.0100349

República de Colombia. (2009). *Ley 1273 de 2009. Por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado - denominado "de la protección de la*

información y de los datos" y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones. Diario Oficial No. 47.223. https://www.sic.gov.co/recursos_user/documentos/normatividad/Ley_1273_2009.pdf

Singh, R., Kumar Gupta, M., Patil, D. R., & Patil, S. M. (2024). Analysis of web application vulnerabilities using dynamic application security testing. En *Proceedings of the 2024 IEEE 9th International Conference for Convergence in Technology (I2CT)* (pp. 1–6). IEEE. https://doi.org/10.1109/I2CT61223.2024.10543484

Somi, V. (2024). A comparative analysis and benchmarking of dynamic application security testing (DAST) tools. *Journal of Engineering and Applied Sciences Technology, 6*(2), 1–6. https://doi.org/10.47363/jeast/2024(6)E139

Tamayo y Tamayo, M. (2011). *El proceso de la investigación científica* (5.ª ed.). Limusa.

Tenable. (2023). *2023 Threat Landscape Report*. https://www.tenable.com/cyber-exposure/2023-threat-landscape-report

Thool, A., & Brown, C. (2025). *Integrating DAST in Kanban and CI/CD: A real-world security case study*. *arXiv*. https://doi.org/10.48550/arXiv.2503.21947

Apéndices

Apéndice A

Glosario

DAST: (Dynamic Application Security Testing) es una técnica de pruebas de seguridad de aplicaciones web que se utiliza para identificar vulnerabilidades en tiempo real mientras la aplicación se está ejecutando (Singh et al., 2024; Somi, 2024).

DevOps: Se trata de una práctica que prioriza la colaboración entre los desarrolladores y los administradores de la operación o infraestructura para desarrollar e implementar software de manera ágil (Thool & Brown, 2025).

Escaneo: Es una técnica automatizada utilizada para buscar y analizar los sistemas, redes, aplicaciones o dispositivos en busca de vulnerabilidades de seguridad y debilidades potenciales (Lyon, 2009; Pooniya & Kumar, 2019).

Explotar: Aprovechar una vulnerabilidad o debilidad en un sistema, red o aplicación para obtener acceso no autorizado o realizar acciones maliciosas en el sistema comprometido (Penetration Testing Execution Standard, 2023).

Fuzzing: Es una técnica de pruebas de seguridad que se utiliza para identificar vulnerabilidades en aplicaciones y sistemas informáticos mediante la generación automatizada de entradas de prueba aleatorias y malformadas (Open Information Systems Security Group, s.f.).

Inyección de SQL: Un ataque que utiliza la entrada del usuario para manipular una base de datos y obtener acceso no autorizado a información confidencial (OWASP Foundation, 2023).

Malware: Software malicioso diseñado para dañar, interferir con, o tomar control de un sistema sin el consentimiento del usuario (República de Colombia, 2009).

OWASP (Open Web Application Security Project): Es una organización sin fines de lucro que se dedica a mejorar la seguridad de las aplicaciones web y la infraestructura de internet (OWASP Foundation, 2023).

Phishing: Un ataque cibernético que utiliza correos electrónicos falsos o sitios web para engañar a las personas y obtener información confidencial, como contraseñas o información financiera (Tenable, 2023).

SAST (Static Application Security Testing): Es una técnica de pruebas de seguridad utilizada para evaluar la seguridad del código fuente de una aplicación antes de que sea implementado en un entorno de producción (López Álvarez, 2020).

SDLC (Software Development Life Cycle): Este ciclo muestra las fases que intervienen en el desarrollo del software desde la planificación hasta el mantenimiento (López Álvarez, 2020).

Sistema Productivo: Se denomina que un sistema se encuentra productivo cuando ya está desplegado en un ambiente de producción y está siendo usado por los usuarios finales (NIST, 2008).

Vulnerabilidad: Es una falla en la seguridad cibernética que puede ser aprovechada por una persona malintencionada para acceder ilegalmente a datos o sistemas (FIRST, 2019; Koman & Janiszewski, 2025).

Apéndice B

Variables Clave

Variable	Definición Operativa	Justificación
Vulnerabilidades que se pueden detectar	Fallos de seguridad que pueden ser identificados mediante pruebas DAST, como inyecciones SQL, XSS, fallas de autenticación, entre otros.	Son el objetivo principal del proceso DAST; su detección es crucial para prevenir ataques antes del paso a producción.
Técnicas para detectar vulnerabilidades	Métodos o estrategias empleadas en las pruebas DAST, como fuzzing, escaneo automatizado o inyección de entradas maliciosas.	Determinan la eficacia del proceso DAST. Su correcta aplicación permite ampliar el espectro de detección de fallos en una aplicación web.
Herramientas para detectar vulnerabilidades	Software o plataformas utilizadas en la ejecución de pruebas DAST, tales como OWASP ZAP, Burp Suite, Acunetix, entre otras.	Permiten implementar las técnicas de detección de forma automatizada y eficiente. Su elección impacta directamente en la calidad del análisis.
Impacto de las vulnerabilidades explotadas	Consecuencias que pueden generarse si una vulnerabilidad es aprovechada por un atacante (pérdida de datos, acceso no autorizado, interrupción de servicio, etc.).	Justifica la necesidad de realizar pruebas DAST. El análisis del impacto permite priorizar la mitigación de vulnerabilidades críticas.
Tiempo de ejecución de las pruebas	Duración necesaria para ejecutar un ciclo completo de pruebas DAST sobre una aplicación o servicio web en un entorno de prueba.	Afecta la viabilidad de integrar DAST en ciclos de desarrollo ágiles. Una ejecución eficiente facilita su adopción dentro del flujo DevSecOps.
Tipos de fallas que se pueden detectar	Clasificación de los errores de seguridad detectables con DAST: errores de validación de entradas, fallos de sesión, configuración insegura, etc.	Permite conocer el alcance del proceso DAST. Analizar estos tipos ayuda a determinar la cobertura de seguridad que puede alcanzarse con este enfoque de prueba.

Nota. La tabla muestra las variables clave que se definieron operativamente a partir de su función dentro del enfoque metodológico y del problema planteado en esta monografía.

Apéndice C

Reporte de Vulnerabilidades Detectadas con ZAP en el Sitio OWASP Juice Shop

Sites: http://cdnjs.cloudflare.com http://127.0.0.1:3000

Generated on Wed, 22 Oct 2025 01:10:18

ZAP Version: 2.16.1

ZAP by Checkmarx

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	4
Low	7
Informational	4
False Positives:	0

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	11
Cross-Domain Misconfiguration	Medium	11
Missing Anti-clickjacking Header	Medium	8
Session ID in URL Rewrite	Medium	12
Cross-Domain JavaScript Source File Inclusion	Low	10

Nota. La imagen muestra la estructura visual del reporte de vulnerabilidades generado por la herramienta OWASP ZAP. El reporte en su versión completa se encuentra disponible para su consulta y descarga en el repositorio en línea:

https://1drv.ms/u/c/1daf2e877e3dd803/EW7onDjHEhRGliGfep5sC9kBVs_QKYbrofSzp3wFG6_TXg?e=IhUsPG

Apéndice D

Formatos de Informes

Informe Técnico de Resultados DAST

Aplicación analizada	
Fecha de análisis	
Motivo de análisis	(Periódico/nuevo despliegue/re-escaneo)
Quién realiza el análisis	
Resultado del análisis	

1. Informe Técnico Detallado

Vulnerabilidad	Severidad	URL afectada	Evidencia técnica	Recomendación	Estado	Documentación a la vulnerabilidad
XSS reflejado	Alta	/search?q=<script>	Carga script malicioso en respuesta	Validar y sanitizar entradas del usuario	Abierta/Cerrada	

Nota. La imagen muestra la estructura visual y la distribución de los componentes para el reporte técnico de vulnerabilidades. Los formatos editables en su versión completa se encuentran disponibles para su consulta y descarga en el repositorio en línea:

https://1drv.ms/w/c/1daf2e877e3dd803/ES0k6S-qpItPk9T1OUVBR3gB3qe6NVZMIAMD0O_suTU-NQ?e=adDBjx