Herramientas para pruebas automatizadas de seguridad en aplicaciones Web: Un mapeo sistemático de la literatura

Elizabeth Gamboa, Christian Quesada-López, Alexandra Martínez, Marcelo Jenkins

Universidad de Costa Rica, San Pedro, Costa Rica
{elizabeth.gamboa, cristian.quesadalopez, alexandra.martinez, marcelo.jenkins}@ucr.ac.cr

Resumen Las herramientas utilizadas para automatizar las pruebas de seguridad en aplicaciones Web son esenciales para detectar vulnerabilidades y prevenir ataques cibernéticos. En este estudio identificamos herramientas reportadas entre el 2006 y el 2019 para probar la seguridad de aplicaciones Web. Cada una de las herramientas es clasificada en términos de los tipos de vulnerabilidades que prueban. Para ello, realizamos un mapeo sistemático de la literatura en el que se analizaron 63 estudios primarios, de los cuales identificamos 66 herramientas utilizadas para realizar pruebas automatizadas de seguridad. Las herramientas se clasificaron según los tipos de la metodología de pruebas de seguridad para determinar vulnerabilidades del proyecto abierto de seguridad en aplicaciones Web *(OWASP)*. La categoría de pruebas para detectar vulnerabilidades más común fue la de *Input Validation Testing (4.8)* con 55 herramientas, seguido de las pruebas de *Configuration and Deployment Management Testing (4.3)*, *Session Management Testing (4.7)*, y *Client Side Testing (4.12)* con 15 herramientas utilizadas cada una. Los tipos de pruebas más reportados fueron los de la categoría *Input Validation Testing (4.8)*. En este caso *SQL Injection (4.8.5)* con 40 herramientas, *Cross-Site Scripting (4.8.2)* con 30 herramientas, y *Testing for HTTP Incoming Requests (4.8.17)* con 19 herramientas utilizadas.

Palabras clave: Pruebas de seguridad, herramientas, aplicaciones Web, vulnerabilidades, OWASP, mapeo sistemático.

1. Introducción

Las pruebas de seguridad para aplicaciones Web son esenciales para prevenir la explotación de vulnerabilidades que generalmente buscan comprometer o dañar un sistema y la información que este administra. Se estima que más del 90 % de las aplicaciones Web son vulnerables, con una media de más de 10 vulnerabilidades por aplicación [1]. Una prueba de seguridad es un método para evaluar la seguridad de un sistema Web mediante la validación y verificación metódica de la efectividad de los controles de seguridad de la aplicación [2]. Una prueba de seguridad Web se centra en evaluar la seguridad de una aplicación

Web, cuyo proceso implica un análisis activo de dicha aplicación en busca de debilidades, fallas técnicas o vulnerabilidades [2]. Para proteger las aplicaciones Web es necesario identificar y eliminar las vulnerabilidades que estas presentan. Una vulnerabilidad es una falla o debilidad en el diseño, implementación, operación o administración de un sistema que podría explotarse para comprometer dicho sistema [2]. Las herramientas de pruebas automatizadas pueden complementar los procesos de pruebas manuales para brindar mayor confiabilidad en la cobertura y para reducir el tiempo de ejecución de los casos de prueba de seguridad. Asimismo, pueden apoyar a los equipos de desarrolladores en la ejecución de estos tipos de pruebas. Por otro lado, estas herramientas aun cuentan con algunas limitantes y retos asociados que deben ser estudiados para así poder incrementar el valor agregado que pueden ofrecer a los equipos de calidad [3,4].

El proyecto abierto de seguridad de aplicaciones Web *(OWASP)* se dedica a promocionar el desarrollo de aplicaciones confiables. Este provee recursos abiertos sobre herramientas, documentos y capítulos para cualquier interesado en mejorar la seguridad de las aplicaciones [5]. El *OWASP* detalla los lineamientos para pruebas de seguridad en el cual lista las vulnerabilidades que las pruebas de seguridad Web deben evaluar [5].

En los últimos años, múltiples estudios sobre el uso de herramientas que permiten generar y ejecutar pruebas automatizadas para evaluar la seguridad de las aplicaciones Web han sido reportadas en la literatura [6]. Estas herramientas permiten conocer las vulnerabilidades que sufren las aplicaciones Web para la protección de los datos que administran. El objetivo de las herramientas es habilitar mecanismos para mejorar la confiabilidad y la seguridad de las aplicaciones que prueban.

El objetivo de esta investigación es identificar las herramientas que han sido utilizadas para probar de forma automatizada la seguridad de aplicaciones Web. Las herramientas se clasificaron según los tipos de la metodología de pruebas de seguridad para determinar vulnerabilidades del proyecto abierto de seguridad en aplicaciones Web [5]. Para lograr el objetivo realizamos un mapeo sistemático de la literatura en el que se analizaron 63 estudios primarios, de los cuales identificamos 66 herramientas utilizadas para realizar pruebas automatizadas de seguridad.

El resto del artículo se estructura de la siguiente manera. La sección 2 menciona las vulnerabilidades de las aplicaciones Web. La sección 3 presenta los trabajos relacionados. La sección 4 explica la metodología del mapeo de literatura. La sección 5 analiza los resultados. La sección 6 presenta las discusiones de este estudio. Finalmente, la sección 7 presenta las conclusiones.

2. Marco Teórico

El proyecto abierto de seguridad de aplicaciones Web *(OWASP)* [7] provee información sobre la seguridad de las aplicaciones Web y periódicamente actualiza listas de los ataques de seguridad más comunes. Asimismo, proporciona documentación técnica sobre estas vulnerabilidades que permite implementar mecanismos de seguridad que prevengan ataques que exploten dichas vulnerabilidades. En el 2017, el *OWASP* reportó los 10 riesgos más críticos para las

aplicaciones Web [8] los cuales indicaron que las fallas de inyección (A1:2017), como SQL, NoSQL, OS o LDAP es la principal vulnerabilidad. Los siguientes riesgos fueron la pérdida de autenticación en las funciones de la aplicación relacionadas a autenticación y gestión de sesiones (A2:2017), la exposición de datos sensibles en las aplicaciones Web y APIs que no los protegen adecuadamente (A3:201), las entidades externas XML (XXE) donde los procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML (A4:2017), la pérdida de control de acceso y las restricciones sobre lo que los usuarios autenticados pueden hacer (A5:2017), la configuración de seguridad incorrecta (*ad hoc* o por omisión) (A6:2017), la secuencia de comandos en sitios cruzados (XSS) con datos no confiables enviados al navegador Web sin una validación y codificación apropiada (A7:2017), la deserialización insegura cuando una aplicación recibe objetos serializados dañinos que son manipulados o borrados por el atacante (A8:2017), los componentes con vulnerabilidades conocidas que se ejecutan con los mismos privilegios que la aplicación (A9:2017), y finalmente el registro y monitoreo insuficientes con la falta de respuesta ante incidentes (A10:2017).

OWASP mantiene una metodología de pruebas de seguridad para determinar vulnerabilidades de seguridad en aplicaciones Web [5]. Esta metodología en su sección 4 detalla 11 categorías donde para cada una lista el conjunto de vulnerabilidades asociadas que se deben evaluar en las aplicaciones Web: *Information Gathering (4.2)* que detalla 10 tipos de vulnerabilidades, *Configuration and Deployment Management Testing (4.3)* que detalla 9 tipos de vulnerabilidades, *Identity Management Testing (4.4)* que detalla 5 tipos de vulnerabilidades, *Authenticaion Testing (4.5)* que detalla 10 tipos de vulnerabilidades, *Authorization Testing (4.6)* que detalla 4 tipos de vulnerabilidades, *Session Management Testing (4.7)* que detalla 8 tipos de vulnerabilidades, *Input Validation Testing (4.8)* que detalla 17 tipos de vulnerabilidades, *Error Handling (4.9)* que detalla 2 tipos de vulnerabilidades, *Weak Cryptography (4.10)* que detalla 4 tipos de vulnerabilidades, *Business Logic Testing (4.11)* que detalla 9 tipos de vulnerabilidades, *Client Side Testing (4.12)* que detalla 12 tipos de vulnerabilidades. A partir de estas categorías, la clasificación de vulnerabilidades para las aplicaciones Web permite enfocar esfuerzos para estudiar herramientas y pruebas que permitan automatizar la identificación de dichas vulnerabilidades.

3. Trabajo Relacionado

Distintos estudios han discutido la seguridad del software en la literatura. En el 2003, Thompson [9] estudió por qué las pruebas de seguridad son difíciles. El autor planteó que no se puede cubrir el cien por ciento del software y que es importante contar con herramientas para pruebas de seguridad automatizadas que mejoren la eficiencia con respecto a la ejecución de pruebas manuales.

Por su parte en el 2006, Curphey y Arawo [10] listaron un conjunto de herramientas para pruebas de aplicaciones Web e indicaron que no existen herramientas que evalúen todas las vulnerabilidades. Los autores indicaron la importancia de contar con diferentes herramientas para evaluar la seguridad, cada una enfocada en distintas fases del ciclo de vida de desarrollo y para distintos tipos de

pruebas tales como pruebas de base de datos, de los *web services*, en *runtime*, de *proxy*, para análisis de código, entre otras.

En el 2010, Pfleeger y Cunningham [11] estudiaron la dificultad de medir la seguridad de las aplicaciones y las estrategias para lograr medir los objetivos de seguridad. Los autores indican que la dificultad es determinar métricas efectivas que permitan una rigurosa y práctica medición de la seguridad.

En el 2015, Rafique et al. [6] identificaron soluciones disponibles para las vulnerabilidades de las aplicaciones Web mediante un mapeo de literatura. Para esto clasificaron las soluciones de acuerdo con la lista de los 10 tipos de vulnerabilidades reportadas por *OWASP* en el 2013. Asimismo, identificaron la fase del ciclo de vida de desarrollo en las que propone una solución para las vulnerabilidades.

Mohammed et al. [12] identificaron 54 enfoques de pruebas de seguridad utilizadas en las fases del ciclo de vida de desarrollo del software (SDLC) mediante un mapeo de literatura en el 2017. Estos enfoques se basan principalmente en análisis estático y análisis dinámico de los artefactos de software. En el 2018, Jafari y Rasoolzadegan [13] exploraron estudios que realizan la definición de patrones de seguridad para las aplicaciones de software. Los autores determinaron que la utilización de los patrones de seguridad ha ido creciendo en las nuevas metodologías de desarrollo o la mejora de las existentes. Este estudio complementó estudios secundarios previos sobre patrones de seguridad [14,15].

En nuestro trabajo realizamos una clasificación de herramientas según los tipos de la metodología de pruebas de seguridad para determinar vulnerabilidades del proyecto abierto de seguridad en aplicaciones Web [5] y cuantificamos las vulnerabilidades más reportadas por los estudios identificados. Realizamos un mapeo de las herramientas desde el 2006 hasta el 2019, lo que proporciona una lista de herramientas actualizada que complementa los estudios previos.

4. Metodología

El mapeo sistemático de la literatura se realizó siguiendo los lineamientos establecidos por Petersen, Vakkalanka y Kuzniarz [16] y las recomendaciones de Kitchenham y Charters [17].

El objetivo de este estudio, formulado con el modelo *Goal Question Metric (GQM)* [18] fue *analizar* las herramientas utilizadas para realizar pruebas automatizadas de seguridad *con el propósito de* caracterizarlas *con respecto a* las vulnerabilidades de seguridad que prueban *desde el punto de vista de* los investigadores *en el contexto* de aplicaciones Web. Para guiar este estudio, se definió la siguiente pregunta de investigación: (RQ) ¿Cuáles herramientas se han utilizado para automatizar las pruebas de seguridad de aplicaciones Web?

4.1. Estrategia de búsqueda y proceso de selección de estudios

Primero se realizó una búsqueda exploratoria para identificar estudios relevantes, que fueron usados como artículos de control. Los artículos de control seleccionados [S01, S02, S03] fueron usados como base en la definición y validación del protocolo de investigación. Estos artículos presentan herramientas para la detección de vulnerabilidades en aplicaciones Web y se seleccionaron por contar con la información necesaria para responder la pregunta de investigación.

A partir del objetivo planteado, las preguntas de investigación, y los términos clave extraídos del título y del resumen de los artículos de control, se construyó la versión inicial de la cadena de búsqueda. Para esto se utilizó el modelo PICO (Población, Intervención, Comparación, Salidas) en el proceso de construcción. La cadena final, que se muestra a continuación, fue producto de un proceso de refinamiento que incluyó varias pruebas piloto para reducir el ruido.

("automat*" OR "tool") AND ("secur* test*") AND ("web*")

Las búsquedas automatizadas se realizaron en las bases de datos *SCOPUS*, *IEEE Xplore*, y *Web of Science*. Se realizó la búsqueda en el título, el resumen y las palabras clave de los artículos. El protocolo del mapeo se desarrolló de Marzo a Mayo del 2019, y la búsqueda automatizada se realizó en Junio del 2019. Los estudios se analizaron entre Junio y Diciembre del 2019.

El número de estudios recuperado para cada base de datos fue de 159 artículos en *SCOPUS*, 81 artículos en *IEEE Xplore*, y 20 artículos en *Web of Science*.

En total se obtuvieron 260 artículos de los cuales 75 eran duplicados, tal como se presenta en la Figura 1.

Los artículos fueron tabulados en MS Excel para los procesos de selección, evaluación y extracción de datos. Se eliminaron los duplicados, se aplicaron los criterios de inclusión y exclusión (I/E) y finalmente se hizo la extracción y el análisis de los resultados. Tras la eliminación de duplicados, se obtuvo un conjunto de 185 estudios.

El proceso de I/E se hizo con base en el título y el resumen de los artículos (cuando hubo duda, se hizo lectura completa del artículo). Se excluyeron publicaciones donde no se encontraba disponible el texto completo (E1) y los artículos secundarios y terciarios (E2). Se incluyeron publicaciones de artículos que trataran sobre herramientas utilizadas para pruebas de seguridad en aplicaciones Web (I1) y solo los artículos en idioma inglés (I2).

A partir de la estrategia de búsqueda y el proceso de selección, se obtuvo un total de 63 artículos. En la Figura 1 detallamos el resultado del proceso de búsqueda y selección de artículos.

La lista completa de los estudios relevantes identificados puede consultarse en el enlace: `https://tinyurl.com/w9e48g3`. Adicional a la información de los artículos y su clasificación de acuerdo a las vulnerabilidades para aplicaciones Web, en este enlace se encuentran los detalles de las herramientas (nombre de la herramienta, lenguaje de programación, entre otros), el grado de automatización de las pruebas de seguridad (automatizadas, semi-automatizadas o manuales), y los tipos de pruebas que realiza cada herramienta. Finalmente, se encuentran los detalles de la evaluación de calidad por artículo, el cual describimos a continuación.

4.2. Evaluación de calidad

La evaluación de la calidad de los estudios se realizó para determinar el nivel de detalle ofrecido sobre los aspectos de interés del análisis y basado en la pregunta de investigación. Estos permiten obtener un *ranking* de la completitud de

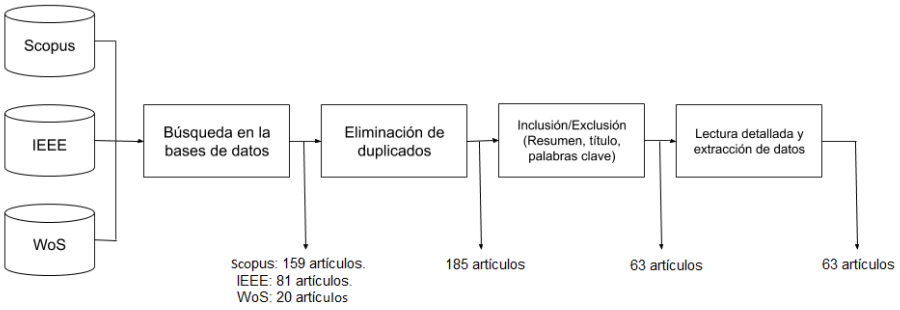

Figura 1: Proceso de búsqueda y selección de artículos

las publicaciones para responder la pregunta de investigación. Los criterios de calidad establecidos para la evaluación de los artículos fueron los siguientes: (Q1) El artículo describe la construcción y la funcionalidad de la herramienta para realizar las pruebas de seguridad. (Q2) El artículo describe cómo se ejecutan las pruebas automatizadas de seguridad utilizando la herramienta. (Q3) El artículo detalla el procedimiento de evaluación de la efectividad de la herramienta utilizada y sus resultados. El puntaje se otorga en una escala de 0 a 2, donde 0 = No en lo absoluto, 1 = Parcialmente y 2= Totalmente.

Los valores de calidad obtenidos por los estudios variaron entre 0 y 6, con una mediana de 4 y un promedio de 3.70, lo que refleja que los estudios tienen un nivel de detalle aceptable. El detalle de la evaluación de calidad por artículo se encuentra disponible en el enlace: `https://tinyurl.com/w9e48g3`.

4.3. Extracción y análisis de los datos

Para la extracción de la información de los 63 artículos se elaboró una tabla con los elementos que permitían responder la pregunta de investigación. Para la pregunta de investigación (RQ) se extrajo la herramienta utilizada y su descripción, el contexto donde se ejecuta la herramienta, la información de las pruebas que realiza para detectar las vulnerabilidades de las aplicaciones Web. Una vez que se contó con la información tabulada se procedió con el análisis de las frecuencias de vulnerabilidades por herramientas y la identificación de las evaluaciones de efectividad de las herramientas. Del mismo modo, el formularios de extracción se encuentra disponible en el enlace: `https://tinyurl.com/w9e48g3`.

4.4. Amenazas a la validez

A continuación se discuten algunas de las limitaciones del estudio. La cadena de búsqueda fue definida a partir de una búsqueda exploratoria en bases de datos y un conjunto de artículos de control. Además fue refinada mediante un conjunto de pruebas piloto. Las bases de datos seleccionadas son reconocidas por tener una buena cobertura de información en el campo de ingeniería de software. Durante el proceso de inclusión o exclusión, si existían dudas sobre un artículo específico, se procedió a su lectura completa.

Las clasificaciones presentadas en este estudio, así como la interpretación de los resultados, se realizaron usando el criterio de la investigadora principal y con la validación de los demás investigadores. Además, la aplicación de los criterios de calidad fue realizada por una sola investigadora. Para la clasificación de las herramientas, se utilizó la metodología de pruebas para aplicaciones Web de *OWASP*. En la mayoría de los casos la clasificación fue extraída de manera explícita de los artículos; sin embargo, para algunos casos la selección de la vulnerabilidad se realizó de manera implícita a partir de los datos reportados. Todo el proceso se reportó de forma detallada para facilitar su análisis y utilización en estudios posteriores.

5. Resultados

A continuación se presentan los resultados del mapeo sistemático de la literatura que identifican las herramientas utilizadas para la automatización de pruebas de seguridad de aplicaciones Web.

En total identificamos 66 propuestas de herramientas utilizadas para realizar pruebas automatizadas de seguridad. Las herramientas se clasificaron según las 11 categorías de la Sección 4 de la metodología de pruebas de seguridad para determinar vulnerabilidades del proyecto abierto de seguridad en aplicaciones Web de *OWASP* (basado en el primer nivel de la clasificación) [5].

El Cuadro 1 agrupa las herramientas para cada una de las categorías de vulnerabilidades del primer nivel de la clasificación de *OWASP*. Las categorías para las cuales se identificaron herramientas fueron: *Configuration and Deployment Management Testing (4.3)*, *Identity Management Testing (4.4)*, *Authenticaion Testing (4.5)*, *Authorization Testing (4.6)*, *Session Management Testing (4.7)*, *Input Validation Testing (4.8)*, *Error Handling (4.9)*, *Weak Cryptography (4.10)* y *Client Side Testing (4.12)*.

Para cada categoría (Id) se presenta la lista de herramientas, referencias de los artículos que la evaluaron, y la cantidad de herramientas (Q). Los resultados indican que la categoría de pruebas para detectar vulnerabilidades más común fue *Input Validation Testing (4.8)* con 55 herramientas, seguido de las pruebas de *Configuration and Deployment Management Testing (4.3)*, *Session Management Testing (4.7)* y *Client Side Testing (4.12)* con 15 herramientas utilizadas cada una.

Para el Cuadro 1 se cuantificó la cantidad de artículos que trabajaron cada categoría de vulnerabilidad y el total de ocurrencias de las vulnerabilidades de una categoría en los artículos. Los resultados muestran una tendencia similar a las de las herramientas donde la categoría de pruebas para detectar vulnerabilidades con más artículos fue la *(4.8)* con 52 artículos y 113 vulnerabilidades evaluadas. En el caso de las pruebas de *(4.3)* la cantidad de artículos y vulnerabilidades evaluadas fue 15. Finalmente, las categorías *(4.7) y (4.12)* fueron reportadas en 12 artículos y en el caso de la categoría *(4.7)* evaluó 12 vulnerabilidades y en la categoría *(4.12)* evaluó 17 vulnerabilidades.

Las herramientas *ZAP* [S58, S37, S42], *ISTA* [S05, S48], *SPaciTE* [S12, S63] y *Volcano* [S49, S52] fueron las únicas reportados por dos o más artículos. Esto

Cuadro 1: Herramientas por categoría de *OWASP* (primer nivel)

Id	Herramientas y referencias	Q
4.3	ISTA [S48], AppScan [S34], Urls black box tests on the Web pages [S32], Fuzz testing tool [S65], HJ2IF [S28], IPT-WS [S54], MobSTer [S04], SAML-based SSO IdP by Google [S64], SECEVAL [S46], WS-Attacker [S33], WSFAgressor [S61], WSInject [S59], WSSecTool [S66], WSAttaker [S57], WSVTS [S58]	15
4.4	SPaCiTE [S12, S63], AppScan [S34]	2
4.5	ZAP [S07, S37], DAST [S37], Fortify [S10], IMAATT [S45], JBroFuzz [S10], OAuthTester [S55], Paros [S10], PBST [S36], Selenium IDE [S20], WebScarab [S10]	10
4.6	ISTA [S05], OAuthTester [S55], SPaCIoS [S19]	3
4.7	ZAP [S07], ISTA [S05], CodePulse [S26], Magento [S14], MobsTer [S04], PBST [S36], SAMATE [S16], Selenium IDE [S20], AOP [S39], Urls black box tests [S32], Deemon [S38], Fortify [S10], JBroFuzz [S10], Paros [S10], WebScarab [S10]	15
4.8	ZAP [S07, S37, S42], ISTA [S05], SPaCiTE [S12, S63], Volcano [S49, S52], AOP [S39], Sign-WS [S54], ATUSA [S51], BIOFUZZ [S43], BurpTool [S42], CRAXweb [S17], Circe [S21], CodePulse [S26], DAST [S37], Deemon [S38], XSS, black box and SQLI injection point [S30], Eclipse IDE [S62], Eclipse IDE test cases [S50], Urls black box tests on the Web pages [S32], Fortify [S10], IAAT [S22], IMATT [S45], JBroFuzz [S10], JWebUnit [S23], JWAST [S40], KamaleonFuzz [S47], Magento [S14], MobSTer [S04], MBT [S41], Noncespace [S06], PHP2XMI [S08], PURITY [S09], Paros [S10], RBVT [S44], RAD-WS [S54], SAFELI [S35], SAMATE [S16], SAML [S64], SAP HANA XS Applications [S56], SECEVAL [S46], SSES [S15], SPaCIoS [S19], SQLIVDT [S18], SQLMAP [S13], Selenium IDE [S20], IDS [S53], Signature evaluation [S24], Tamper data [S60], Tool-prototype [S11], WAP [S35], WSFAgressor [S61], WSInject [S59], WSVTS [S58], WebScarab [S10], XSSINJECTOR [S27], XSS and SQLI [S25]	55
4.9	ZAP [S07]	1
4.10	ISTA [S05], Fortify [S10], JBroFuzz [S10], Paros [S10], SAML-based SSO IdP service provided by Google [S64], WebScarab [S10]	6
4.12	ZAP [S37], ISTA [S52], AOP [S39], ATUSA [S51], Code Pulse [S26], DAST [S37], Fortify [S10], JBroFuzz [S10], JWAST [S40], KITE [S31], Paros [S10], SAP HANA XS Applications [S56], SQTL [S29], WebScarab [S10], XSSINJECTOR [S27]	15

indica la necesidad de que los investigadores realicen replicaciones utilizando propuestas de herramientas existentes.

En la Figura 2 se presenta la tendencia de vulnerabilidades evaluadas por categoría de *OWASP* del primer nivel por cada año desde el 2006 hasta el 2019. Los resultados confirman que la categoría de vulnerabilidad *Input Validation Testing (4.8)* es la más evaluada por las herramientas entre los años 2007 al 2019, siendo el año 2014 y 2018 el más evaluado con 21 y 17 ocurrencias respectivamente.

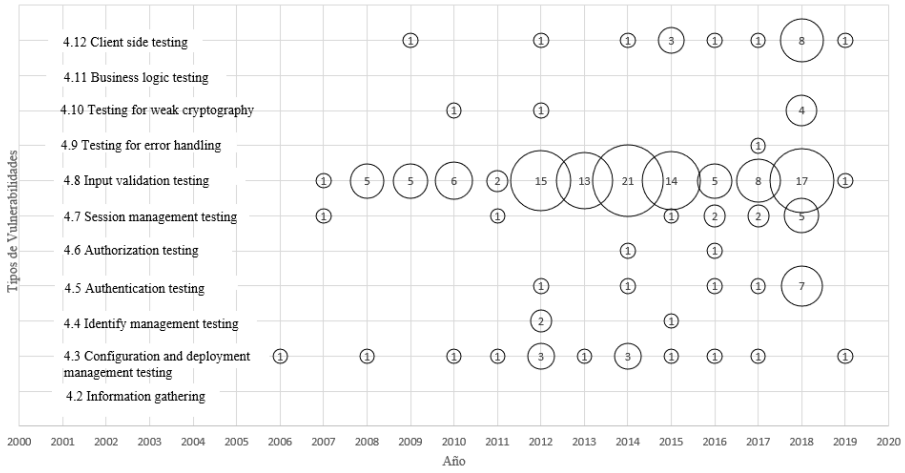

Figura 2: Tendencia de vulnerabilidades del primer nivel de *OWASP* por año

El Cuadro 2 presenta las vulnerabilidades del segundo nivel de *OWASP* para las categorías en que se identificaron herramientas. Las vulnerabilidades para las cuales se identificaron herramientas fueron: *Testing for Privilege Escalation (4.6.3)*, *Testing for Insecure Direct Object References (4.6.4)*, *Testing for Cookies attributes (4.7.2)*, *Testing for Cross Site Request Forgery (CSRF) (4.7.5)*, *Testing for Reflected Cross Site Scripting (4.8.1)*, *Testing for Stored Cross Site Scripting (4.8.2)*, *Testing for SQL Injection (4.8.5)*, *Testing for LDAP Injection (4.8.6) Testing for XML Injection (4.8.8) Testing for XML injection (4.8.9) Testing for XPath Injection (4.8.10) Testing for Command Injection (4.8.13) Testing for HTTP Incoming Requests (4.8.17) Testing for DOM based Cross Site Scripting (4.12.1) Testing for JavaScript Execution (4.12.2) Testing for HTML Injection (4.12.3).*

Para cada vulnerabilidad (Id) se presenta la lista de herramientas y referencias de los artículos que la evaluaron, y la cantidad de herramientas (Q). Los tipos de pruebas más reportadas fueron los de la categoría *Input Validation Testing (4.8)* en la que se encontraron las vulnerabilidades más reportadas. Las vulnerabilidades más comúnmente evaluadas fueron la *SQL Injection (4.8.5)* con 40 herramientas, *Cross-Site Scripting (4.8.2)* con 30 herramientas, y *Testing for HTTP Incoming Requests (4.8.17)* con 19 herramientas utilizadas.

En la Figura 3 se presenta la tendencia de vulnerabilidades evaluadas del segundo nivel de *OWASP* por cada año desde el 2007 hasta el 2019. Los re-

sultados confirman que las vulnerabilidades *Testing for SQL Injection (4.8.5)* y *Cross-Site Scripting (4.8.2)* son las más evaluadas por las herramientas.

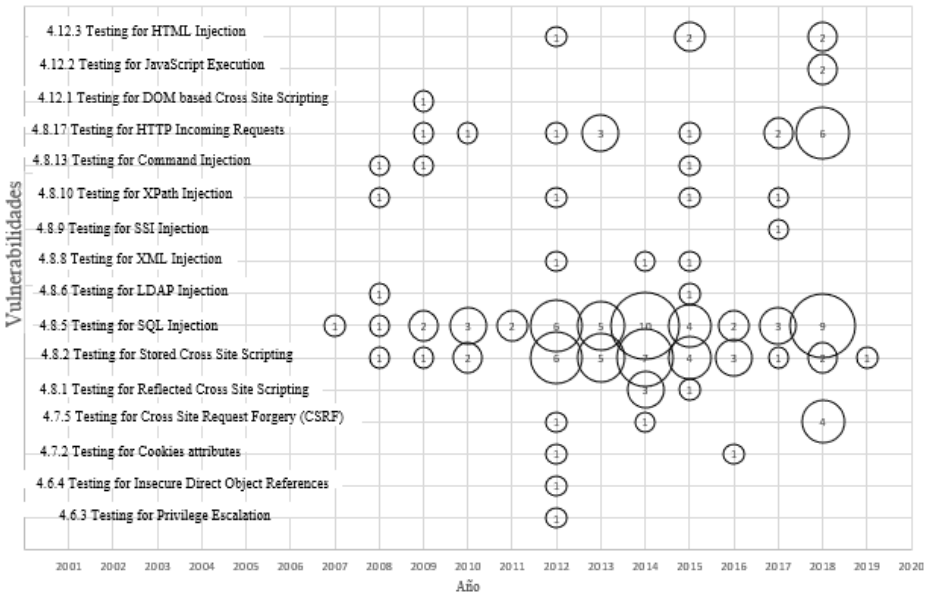

Figura 3: Tendencia de vulnerabilidades del segundo nivel de *OWASP* por año

6. Discusión

La clasificación de las distintas herramientas disponibles para evaluar cada una de las vulnerabilidades de las aplicaciones Web puede apoyar a los profesionales en la selección de potenciales herramientas para sus procesos de pruebas de seguridad. Estas herramientas pueden apoyar la evaluación de la seguridad de las aplicaciones Web de forma automatizada en distintas fases del ciclo de vida de desarrollo de las aplicaciones Web o inclusive cuando se encuentra en producción.

Se identificó gran variedad de herramientas que se utilizan para automatizar distintas pruebas de seguridad de acuerdo con las distintas vulnerabilidades que existen en la actualidad. Para evaluar la seguridad de las aplicaciones Web, las herramientas deben de contar con pruebas actualizadas de acuerdo con las vulnerabilidades y ataques cibernéticos actuales. La identificación de herramientas demostró que se mantiene la tendencia de realizar pruebas para la *SQL Injection* y *Cross-Site Scripting*. Sin embargo, no se encontraron herramientas para algunos de los escenarios reportados como los 10 ataques con más riesgosos y frecuentes por *OWASP*, tales como: *Insecure Deserialization, Using Components with known vulnerabilities* e *Insufficient Logging & Monitoring*.

Por otro lado, de las 66 propuestas de herramientas utilizadas para realizar pruebas automatizadas de seguridad solo 41 reportaron una evaluación de la efectividad. De las 41 herramientas, 16 fueron evaluadas a partir de criterios de

Cuadro 2: Herramientas por vulnerabilidad de *OWASP* (segundo nivel)

Id	Herramientas y referencias	Q
4.6.3	ISTA [S48]	1
4.6.4	ISTA [S48]	1
4.7.2	AOP [S39], Urls and performing black box tests on the Web pages [S32]	2
4.7.5	ISTA [S05], Deemon [S38], Fortify [S10], JBroFuzz [S10], Paros [S10], WebScarab [S10]	6
4.8.1	ZAP [S07], BurpTool [S42], MBT [S41], SPaCIoS [S19]	4
4.8.2	ZAP [S07, S37, S42], ISTA [S05], SPaCiTE [S12, S63], AOP [S39], BurpTool [S42], CRAXweb [S17], Circe [S21], CodePulse [S26], DAST [S37], XSS, black box and SQLI based on injection point [S30], IAAT [S22], IMAATT [S45], JwebUnit [S10], JWAST [S40], KamaleonFuzz [S47], MobSTer [S04], MBT [S04], Noncespaces [S06], PHP2XMI [S08], PURITY [S09], RBVT [S44], SSES [S15], SPaCIoS [S19], SQLIVDT [S18], Selenium IDE [S20], Tamper data [S60], Tool-prototye [S11], WSInject [S59], XSSINJECTOR [S27], XSS and SQLI evaluation [S25]	30
4.8.5	ZAP [S07,S37], ISTA [S05], SPaCiTE [S12, S63], Volcano [S49, S52], AOP [S39], BIOFUZZ [S43], CRAXweb [S17], Circe [S21], CodePulse [S26], DAST [S37], XSS, black box and SQLI based on injection point [S30], Eclipse IDE [S62], Fortify [S10], IAAT [S22], IMAATT [S45], JBroFuzz [S10], JWAST [S40], Magento [S14], MobSTer [S04], PHP2XMI [S08], PURITY [S09], Paros [S10], RBVT [S44], RAD-WS [S54], Eclipse IDE test cases execution [S50], SAFELI [S35], SAMATE [S16], SAP HANA XS Applications [S56], SSES [S15], SPaCIoS [S19], SQLIVDT [S18], SQLMap [S13], Selenium IDE [S20], IDS [S53], Signature evaluation [S24], Tamper data [S60], Tool-prototype [S11], WAP [S35], WebScarab [S10], XSS and SQLI security evaluation [S25]	40
4.8.6	JWAST [S40], SSES [S15]	2
4.8.8	Eclipse IDE [S62], JWAST [S40], SPaCioS [S19]	3
4.8.9	Sign-WS [S54]	1
4.8.10	Eclipse IDE [S62], JWAST [S40], RAD-WS [S54], SSES [S15]	4
4.8.13	JWAST [S40], SSES [S15]	2
4.8.17	ZAP [S07, S37, S42], ATUSa [S51], BurpTool [S42], CRAXweb [S17], DAST [S37], Deemon [S38], XSS, black box and SQLI based on injection point [S30], Urls and performing black box tests on the Web pages [S32], Fortify [S10], JBroFuzz [S10], JWAST [S40], Paros [S10], SAML-based SSO IdP service by Google [S64], SECEVAL [S46], SPaCIoS [S19], Tamper data [S60], WSFAgressor [S61], WSVTS [S58], WebScarab [S10]	19
4.12.1	ATUSA [S51]	1
4.12.2	ZAP [S37], DAST [S37], JWAST [S40], SAP HANA XS Applications [S56]	4
4.12.3	ZAP [S37], ISTA [S05], DAST [S37]	3

eficiencia, 13 a partir de criterios de eficacia y 12 fueron evaluadas a partir de ambos criterios. Las herramientas fueron evaluadas para determinar el tiempo que toma la ejecución de las pruebas (por ejemplo: [S07, S13, S34, S54, S61]), comparar la efectividad entre herramientas (por ejemplo: [S04, S30, S58]), y determinar la relación entre eficiencia y efectividad (cantidad de pruebas ejecutadas y tiempo que tomó la ejecución de las pruebas, por ejemplo: [S11, S41, S24, S30, S32, S38, S43, S51]). Por ejemplo, las herramientas [S35, S36, S47, S54, S59, S61, S04, S51] evaluaron la cantidad de falsos positivos en los resultados de ejecución de las pruebas y la herramienta [S10] comparó los resultados obtenidos al ejecutar pruebas automatizadas contra la ejecución de pruebas manual para comparar los resultados obtenidos. En muchas de las evaluaciones se validó el éxito de la ejecución de pruebas a partir de la cantidad de vulnerabilidades encontradas (por ejemplo: [S09, S18, S19, S20, S21, S23, S25, S31, S10, S30, S32, S43, S58, S42]). Otras evaluaciones se limitaron a verificar la funcionalidad implementada por la herramienta (por ejemplo: [S22, S05, S24, S38, S51, S49]).

Del total de herramientas, 29 herramientas reportaron algunos de los retos que enfrentaron durante la ejecución de las pruebas. Diez herramientas plantean la necesidad de actualizar y cambiar sus tipos de pruebas para mejorar el desempeño, la efectividad y la escalabilidad [S05, S18, S41, S47, S52, S58, S59, S60, S62]. Ocho herramientas plantean la necesidad de crear nuevos casos de pruebas para detectar otros tipos de vulnerabilidades [S04, S08, S13, S14, S23, S30, S36, S51, S57]. Una propuesta plantea la necesidad de mejorar la configuración de la herramienta [S09], otra la mejora en la priorización de las pruebas para utilizarlas en pruebas de regresión [S20], cuatro indican la necesidad de optimizar el código, los métodos y las funcionalidades de la herramienta [S34, S38, S45, S46], así como también la mejora en la detección de los falsos positivos en los resultados [S37]. Finalmente, 3 herramientas plantean la necesidad de mejorar el soporte para distintas [S11, S26, S33].

7. Conclusiones

Este estudio reportó los resultados de un mapeo sistemático de la literatura sobre herramientas utilizadas para probar la seguridad de aplicaciones Web. Se identificaron 63 estudios primarios que reportaron 66 propuestas de herramientas utilizadas para realizar pruebas automatizadas de seguridad. Las herramientas se clasificaron según los tipos de la metodología de pruebas de seguridad para determinar vulnerabilidades del proyecto abierto de seguridad en aplicaciones Web.

Los resultados demuestran que se mantiene la tendencia de realizar pruebas para la *SQL Injection* y *Cross-Site Scripting*. Sin embargo, los casos de prueba que realizan las herramientas identificadas solo evalúan algunos de los escenarios reportados como los 10 ataques más riesgosos y frecuentes por *OWASP*. Además, aunque se identificó gran variedad de herramientas que realizan distintos tipos de pruebas de seguridad, solo pocas herramientas se encuentran en la lista de herramientas recomendadas por *OWASP*, lo que denota la necesidad de contar con evaluaciones empíricas de herramientas existentes en el área.

Como trabajo futuro, se desea seleccionar un conjunto de herramientas de pruebas de seguridad que permitan verificar las principales vulnerabilidades reportadas para las aplicaciones Web y evaluar su efectividad con el fin de generar evidencia para la industria. Asimismo, como trabajo futuro se desea identificar las clasificaciones de vulnerabilidades brindadas por *OWASP* que no fueron cubiertas por ninguna herramienta del mapeo y realizar un estudio de la razón por la cual las actuales herramientas no evalúan estas vulnerabilidades y la importancia de que las herramientas ejecute pruebas para estas vulnerabilidades.

Referencias

1. P. Zech, M. Felderer, and R. Breu, "Knowledge-based security testing of web applications by logic programming," *International Journal on Software Tools for Technology Transfer*, vol. 21, no. 2, pp. 221–246, 2019.
2. OWASP, "Testing: Introduction and objectives," *The Open Web Application Security Project*, 2019.
3. S. De Vries, "Security testing web applications throughout automated software tests," *Corsaire Ltd*, vol. 3, 2006.
4. Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. benefits and drawbacks. modern tendency," in *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*. IEEE, 2016, pp. 488–491.
5. OWASP, "Testing guide v4, web application security testing," *The Open Web Application Security Project*, 2019.
6. S. Rafique, M. Humayun, B. Hamid, A. Abbas, M. Akhtar, and K. Iqbal, "Web application security vulnerabilities detection approaches: A systematic mapping study," in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2015, pp. 1–6.
7. OWASP, "The open web application security project," *The Open Web Application Security Project*, 2019.
8. ——, "Top 10–2017–the ten most critical web application security risks," *The Open Web Application Security Project*, 2017.
9. H. H. Thompson, "Why security testing is hard," *IEEE Security Privacy*, vol. 1, no. 4, pp. 83–86, July 2003.
10. M. Curphey and R. Arawo, "Web application security assessment tools," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 32–41, 2006.
11. S. Pfleeger and R. Cunningham, "Why measuring security is hard," *IEEE Security Privacy*, vol. 8, no. 4, pp. 46–54, July 2010.
12. N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Computer Standards & Interfaces*, vol. 50, pp. 107–115, 2017.
13. A. J. Jafari and A. Rasoolzadegan, "Security patterns: A systematic mapping study," *arXiv preprint arXiv:1811.12715*, 2018.
14. M. Bunke, "Software-security patterns: degree of maturity," in *Proceedings of the 20th European Conference on Pattern Languages of Programs*. ACM, 2015, p. 42.
15. Y. Ito, H. Washizaki, M. Yoshizawa, Y. Fukazawa, T. Okubo, H. Kaiya, A. Hazeyama, N. Yoshioka, and E. B. Fernandez, "Systematic mapping of security patterns research," in *Proceedings of the 22nd Conference on Pattern Languages of Programs*. The Hillside Group, 2015, p. 14.

16. K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," vol. 64. Elsevier, 2015, pp. 1–18.
17. B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Technical report, EBSE Technical Report EBSE-2007-01*, pp. 1–57, 2007.
18. V. Basili, C. Gianluigi, and D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.