Análisis de la vulnerabilidad XSS persistente: Estado actual, medidas de mitigación y herramientas de detección

Persistent XSS vulnerability analysis: Current status, mitigation measures and detection tools.

Pinta Higuita Sergio Ivan[1]
sergio.pinta00@usc.edu.co

Ordoñez Serna Farid[1]
farid.ordonez00@usc.edu.co

Javier S. Rojas-Montes[2]
javier.rojas00@usc.edu.co

Estudiantes, Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad Santiago de Cali (1)
Profesor, Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad Santiago de Cali (2)
Santiago de Cali, 2025

Resumen

Este artículo de revisión se centra en la amenaza del Cross-Site Scripting (XSS) persistente en aplicaciones web, desde su surgimiento, evolución, métodos de explotación, nuevas tendencias emergentes y su impacto en el sector informático. Se realizó una revisión sistemática de la literatura siguiendo la metodología PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), seleccionando fuentes relevantes a través de bases de datos académicas como SpringerLink, Taylor & Francis Online, Scopus, Google Scholar, IEEE Xplore, ACM Digital Library y ScienceDirect. Los estudios fueron evaluados según criterios de relevancia, actualidad y calidad de datos. Posteriormente, se revisan herramientas de detección como OWASP ZAP, Burp Suite y Acunetix, destacando su papel en la identificación temprana de vulnerabilidades. También se examinan estrategias de prevención y mitigación, que incluyen la validación rigurosa de datos, la codificación segura, el uso de frameworks y herramientas de seguridad, así como la importancia de la capacitación del personal encargado de la seguridad informática, respecto a la concientización sobre la importancia de la ciberseguridad en las organizaciones. Adicionalmente, se propone la integración de prácticas de seguridad en todo el ciclo de vida del software para garantizar una defensa robusta contra XSS persistente. La información recopilada se analizó y sintetizó para proporcionar una visión completa de la vulnerabilidad XSS persistente. La conclusión más relevante resalta la importancia de una defensa en profundidad mediante herramientas de detección, políticas de seguridad bien definidas y la formación continua de personal, como enfoques críticos para mitigar eficazmente los riesgos asociados a XSS persistente en aplicaciones web.

Palabras clave: XSS persistente, ciberseguridad, mitigación de vulnerabilidades, codificación segura, Análisis de Vulnerabilidad.

Abstract

This review article focuses on the threat of persistent Cross-Site Scripting (XSS) in web applications, covering its emergence, evolution, exploitation methods, emerging trends, and impact on the IT sector. A systematic literature review was conducted following the PRISMA methodology (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), selecting relevant sources from academic databases such as SpringerLink, Taylor & Francis Online, Scopus, Google Scholar, IEEE Xplore, ACM Digital Library, and ScienceDirect. Studies were evaluated based on relevance, timeliness, and data quality. Subsequently, detection tools such as OWASP ZAP, Burp Suite, and Acunetix are reviewed, highlighting their role in the early identification of vulnerabilities. Prevention and mitigation strategies are also examined, including rigorous data validation, secure encoding, the use of frameworks and security tools, and the importance of training IT security personnel to raise awareness of cybersecurity's importance within organizations. Additionally, integrating security practices throughout the software lifecycle is proposed to ensure a robust defense against persistent XSS. The information collected was analyzed and synthesized to provide a comprehensive view of the persistent XSS vulnerability. The most relevant conclusion emphasizes the importance of a defense-in-depth approach, using detection tools, well-defined security policies, and continuous staff training as critical approaches to effectively mitigate the risks associated with persistent XSS in web applications.

Keywords: Persistent XSS, cybersecurity, vulnerability mitigation, secure coding, vulnerability analysis

1. INTRODUCCIÓN

En la era digital actual, las aplicaciones web han transformado la manera en que interactuamos con la información y los servicios en línea, pero también han introducido nuevos desafíos en seguridad cibernética. Una de las amenazas más significativas es el Cross-Site Scripting (XSS) persistente, que ocurre cuando atacantes insertan scripts maliciosos en páginas web, los cuales se almacenan en el servidor y se ejecutan en los navegadores de los usuarios. Esta vulnerabilidad puede comprometer datos sensibles, robar credenciales y, en casos extremos, tomar el control de sesiones de usuarios.

Para abordar este problema, se realizó una revisión sistemática de la literatura siguiendo la metodología PRISMA. Se seleccionaron artículos relevantes de bases de datos académicas como SpringerLink, Scopus y IEEE Xplore, priorizando publicaciones de los últimos diez años que presentaran datos empíricos o casos de estudio. Este enfoque garantiza una base sólida para las conclusiones y recomendaciones.

Los hallazgos indican que las vulnerabilidades de XSS persistente siguen siendo comunes debido a la insuficiente validación de entradas y la falta de conciencia en seguridad cibernética entre los desarrolladores. Herramientas como OWASP ZAP y Burp Suite son efectivas para la detección y mitigación de estas vulnerabilidades. Mientras que OWASP ZAP es una opción popular de código abierto, Burp Suite ofrece funcionalidades avanzadas en un entorno comercial.

Se destaca la necesidad urgente de un enfoque proactivo y multifacético para proteger las aplicaciones web contra XSS persistente, integrando prácticas de seguridad en todo el ciclo de vida del software. Esto incluye la implementación de herramientas de detección y la promoción de una cultura de seguridad entre desarrolladores. La educación continua sobre mejores prácticas de seguridad es esencial para reducir el riesgo de vulnerabilidades.

Finalmente, se recomienda un enfoque holístico que combine tecnologías avanzadas, formación constante y políticas de seguridad efectivas. La colaboración entre equipos dentro de una organización es crucial para proteger los datos sensibles de los usuarios y mitigar los riesgos asociados con el XSS persistente y otras amenazas cibernéticas.

2. METODOLOGÍA

Se empleó una adaptación de la metodología PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) (Page et al., 2021) para analizar la vulnerabilidad de XSS persistente. Esta metodología garantiza un enfoque estructurado y transparente para la selección y análisis de estudios relevantes. A continuación, se detallan los pasos seguidos:

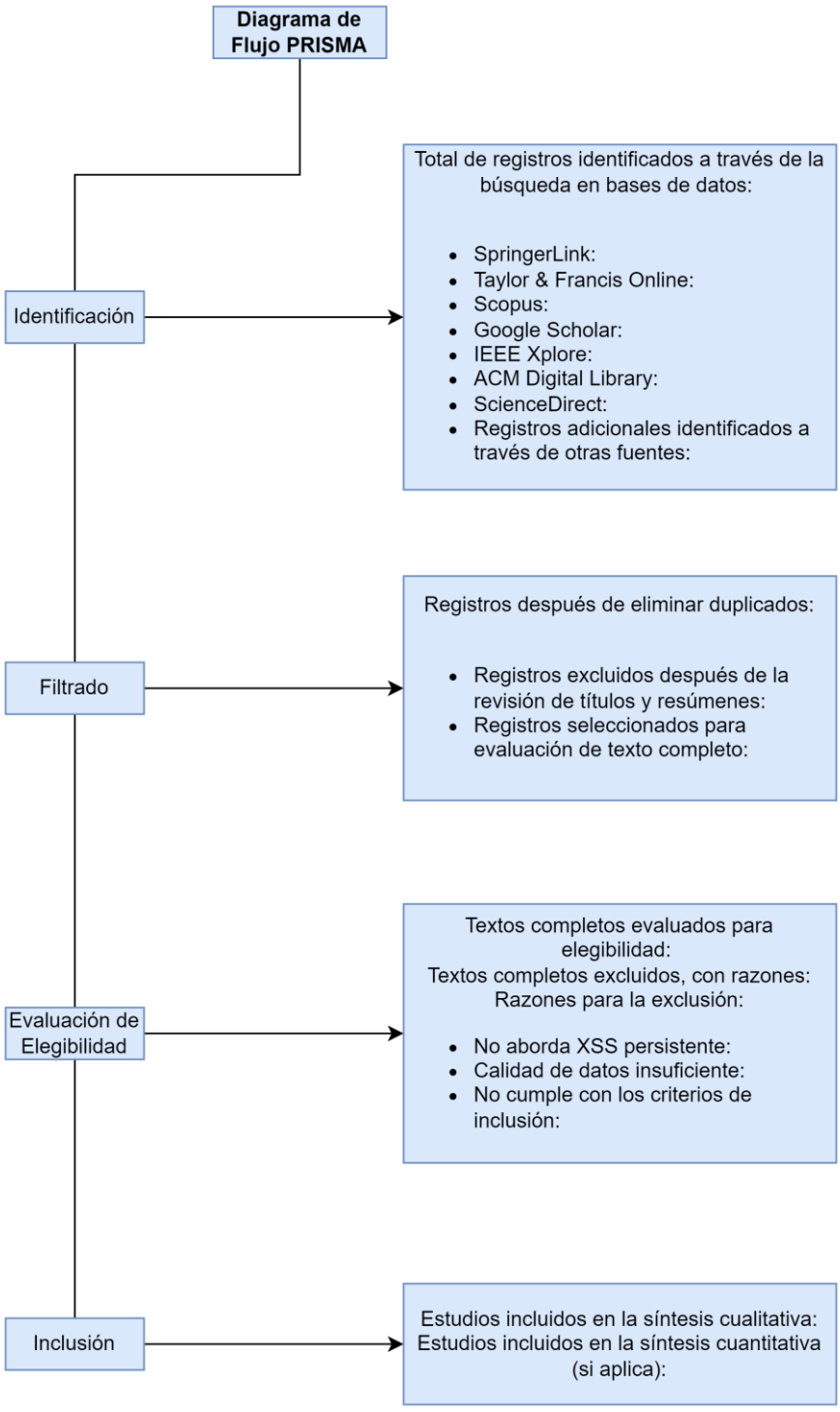


Ilustración 1 Diagrama de Flujo de Selección de Estudios para la Revisión Sistemática sobre XSS Persistente (Elaboración Propia)

2.1. Identificación de Estudios:

- Búsqueda de Fuentes: Se realizaron búsquedas sistemáticas en bases de datos académicas y técnicas, tales como Google Scholar, IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, Taylor & Francis Online, y Scopus.
- Palabras Clave: Se utilizaron combinaciones de términos clave como "XSS persistente", "vulnerabilidad", "mitigación de XSS", "detección de XSS", "ciberseguridad", "ataques de scripts", "prevención de XSS", y "herramientas de seguridad web".

2.2. Selección de Estudios:

2.2.1. Criterios de Inclusión:

- Estudios publicados en los últimos diez años para asegurar la actualidad de la información.
- Publicaciones que aborden específicamente la vulnerabilidad de XSS persistente.
- Artículos que presenten datos empíricos, análisis detallados, revisiones sistemáticas, o casos de estudio relevantes.
- Documentos disponibles en inglés o español.

2.3. Criterios de Exclusión:

- Artículos con datos insuficientes, no fundamentados, o que no aporten información significativa sobre el tema.
- Opiniones sin respaldo empírico o artículos de divulgación sin rigor científico.
- Publicaciones duplicadas o irrelevantes para el enfoque específico del artículo.

2.4. Extracción de Datos:

- Proceso de Selección: Los títulos y resúmenes de los artículos identificados fueron revisados para determinar su relevancia inicial. Aquellos artículos que parecían pertinentes fueron evaluados en detalle mediante la lectura completa del texto.
- Revisión Completa: Los estudios seleccionados fueron leídos y analizados en profundidad para extraer información clave sobre la naturaleza de la vulnerabilidad XSS persistente, métodos de explotación, medidas de mitigación, y herramientas de detección.

2.5. Análisis y Síntesis de Información:

- Agrupación Temática de Datos: La información extraída fue agrupada temáticamente para facilitar la comparación y síntesis. Los temas principales incluyeron: antecedentes y evolución del XSS persistente, métodos de explotación y vectores de ataque, estrategias de mitigación y buenas prácticas, y herramientas de detección y evaluación.
- Síntesis Comparativa: Se realizó una síntesis comparativa de las estrategias de mitigación y herramientas de detección más utilizadas, destacando sus ventajas, limitaciones, y casos de éxito en su implementación.

2.6. Evaluación de Herramientas y Técnicas:

- Evaluación Comparativa: Se incluyó una evaluación comparativa de herramientas destacadas para la detección y mitigación de XSS persistente, tales como OWASP ZAP, Burp Suite, Acunetix, Netsparker, Arachni, y W3AF. Se evaluaron aspectos como precisión, facilidad de uso, coste, y adecuación a diferentes tipos de proyectos y tamaños de empresas.

- Estudios de Casos: Se analizaron casos de estudio relevantes para ilustrar la aplicación práctica de estas herramientas y técnicas en entornos reales, destacando ejemplos significativos de incidentes y cómo fueron mitigados.

3. ANTECEDENTES Y EVOLUCIÓN DEL XSS PERSISTENTE

3.1. Evolución Histórica del XSS Persistente

El XSS persistente es una de las vulnerabilidades más peligrosas y persistentes en las aplicaciones web modernas, resultado del entorno dinámico y en rápido crecimiento de las aplicaciones web desde los años 90 (*Grossman*, s. f.; Pintus Antonio et al., s. f.). En sus inicios, la seguridad no era una prioridad en el desarrollo, y la falta de validación en las interfaces web permitió que los atacantes insertaran scripts maliciosos que permanecían activos en las páginas, afectando a otros usuarios (S. Gupta & Sharma, 2012).

Con el avance de tecnologías como AJAX y frameworks de JavaScript, los métodos de explotación también evolucionaron, permitiendo a los atacantes engañar a los usuarios para ejecutar scripts sin su conocimiento, aumentando así el impacto de los ataques (Heiderich et al., 2012). La industria de la seguridad informática reaccionó desarrollando nuevas herramientas y estableciendo mejores prácticas, como la validación estricta de datos y la implementación de políticas de seguridad (OWASP, 2023).

Este desarrollo resalta la necesidad constante de mitigar amenazas emergentes en ciberseguridad y de adoptar medidas proactivas para proteger las aplicaciones web y la confianza de los usuarios (Kurniawan et al., 2015).

3.1.1. Antecedentes Históricos

El Cross-Site Scripting (XSS) persistente tiene sus raíces en los primeros días del desarrollo de aplicaciones web, cuando la seguridad en línea no era una prioridad absoluta (Thexssrat, 2023). A medida que las tecnologías web evolucionaban y las aplicaciones se volvían más interactivas y dinámicas, surgieron inadvertidamente vulnerabilidades significativas que los atacantes comenzaron a explotar de manera más sofisticada.

3.1.1.1. Orígenes y Primeros Incidentes Documentados

El XSS persistente surgió en la década de 1990, cuando el rápido crecimiento de las aplicaciones web priorizaba la funcionalidad sobre la seguridad. La falta de validación rigurosa de los datos de entrada facilitó la inserción de scripts maliciosos que persistían en la página web y se ejecutaban automáticamente cuando otros usuarios la visitaban (Endler, 2002). Este tipo de ataques comprometía la integridad y confidencialidad de los datos de las aplicaciones y marcó un punto de inflexión en la seguridad web, al evidenciar las graves vulnerabilidades en el manejo de datos por los desarrolladores (Grossman, 2007).

Con el tiempo, la comunidad de seguridad informática reconoció la seriedad de este problema y empezó a desarrollar contramedidas para prevenir el XSS persistente y proteger a los usuarios.

3.1.1.2. Desarrollo y Complejidad Creciente de los Ataques

Con el tiempo, la evolución de las tecnologías web introdujo nuevas capas de complejidad y oportunidades para los ataques de XSS persistente. La adopción generalizada de tecnologías como AJAX y frameworks de JavaScript avanzados permitió a los desarrolladores crear aplicaciones web más dinámicas e interactivas (Hydara et al., 2015). Sin embargo, estas mismas tecnologías también ampliaron la superficie de ataque y ofrecieron a los atacantes nuevas vías para explotar vulnerabilidades.

Los métodos de explotación del XSS persistente se volvieron más sofisticados a medida que los atacantes descubrían formas de eludir las defensas tradicionales (Mohammadi et al., 2016). Por ejemplo, utilizaron técnicas como la codificación de URL y la fragmentación de código malicioso en múltiples solicitudes para evitar la detección y ejecución

de scripts en contextos vulnerables de aplicaciones web.

La respuesta de la industria de la seguridad informática fue rápida y decisiva. Se desarrollaron herramientas avanzadas de detección y mitigación de XSS persistente, y se promovieron mejores prácticas entre los desarrolladores para implementar controles de seguridad desde el diseño inicial de la aplicación. La educación continua sobre la importancia de la validación de entrada de datos y la sanitización adecuada se convirtió en un componente crucial de la formación de desarrolladores y administradores de sistemas.

3.1.2. Desarrollo y Complejidad Creciente de los Ataques

El Cross-Site Scripting (XSS) persistente ha evolucionado significativamente a lo largo de los años, adaptándose a los avances tecnológicos y a las prácticas de desarrollo de aplicaciones web. A medida que las aplicaciones se volvieron más interactivas y dinámicas con el uso extendido de tecnologías como AJAX y frameworks de JavaScript, también aumentaron las oportunidades para los ataques de XSS persistente.

3.1.2.1. Expansión de Superficies de Ataque

Con la adopción generalizada de AJAX (Asynchronous JavaScript and XML), las aplicaciones web comenzaron a realizar más interacciones entre el cliente y el servidor sin recargar la página completa. Esto permitió una experiencia de usuario más fluida pero también introdujo nuevas vulnerabilidades(Bates et al., 2010). Los ataques de XSS persistente pudieron aprovechar estas interacciones dinámicas para introducir y ejecutar scripts maliciosos en contextos que tradicionalmente podrían considerarse seguros.

Además, la popularidad de los frameworks de JavaScript como AngularJS, React y Vue.js introdujo técnicas más avanzadas para la manipulación del DOM (Document Object Model), lo cual también ofreció nuevas oportunidades para los atacantes. Estos frameworks permiten a los desarrolladores crear aplicaciones altamente interactivas, pero al mismo tiempo requieren un manejo cuidadoso de los datos de entrada para prevenir vulnerabilidades de XSS persistente.

3.1.2.2. Sofisticación de Técnicas de Elusión

A medida que las defensas contra el XSS persistente se fortalecieron, los atacantes desarrollaron técnicas más sofisticadas para eludir estas defensas. Por ejemplo, comenzaron a codificar los scripts maliciosos utilizando diferentes métodos de codificación como la codificación Unicode, la codificación de URL o la codificación base64 (Siriwardena, 2019). Estos métodos dificultan la detección automática de scripts maliciosos por parte de los filtros y mecanismos de seguridad implementados en las aplicaciones web.

Otra técnica común utilizada por los atacantes es la fragmentación del código malicioso en múltiples solicitudes HTTP (Ajay Pal Singh & Ashwani Kumar, 2022). Al dividir el código del script en partes más pequeñas y distribuirlo en varias solicitudes, los atacantes pueden evitar la detección inicial y la mitigación de XSS persistente que se centran en bloquear scripts en una única solicitud HTTP.

3.1.2.3. Desafíos Emergentes

Con el aumento del uso de APIs (Application Programming Interfaces) y servicios web en las aplicaciones modernas, también han surgido nuevos desafíos para la seguridad contra el XSS persistente. Las API RESTful (Khan et al., 2024), por ejemplo, pueden transmitir datos entre el cliente y el servidor de manera eficiente pero también pueden introducir nuevas oportunidades para la inyección de scripts maliciosos si no se implementan medidas de seguridad adecuadas.

Además, la convergencia de aplicaciones web y móviles ha ampliado aún más las superficies de ataque potenciales. Las aplicaciones híbridas que combinan componentes web y nativos pueden ser especialmente vulnerables si no se aplican correctamente los principios de seguridad tanto en el lado del cliente como en el servidor.

3.1.2.4. Respuesta de la Industria y Mejores Prácticas

Frente a estos desafíos, la industria de la seguridad informática ha respondido con herramientas avanzadas de detección y mitigación de XSS persistente. También se han establecido directrices y mejores prácticas para desarrolladores y

administradores de sistemas, incluyendo la implementación rigurosa de la validación de entrada de datos y la sanitización adecuada para prevenir este tipo de vulnerabilidad desde el diseño inicial de las aplicaciones.

En resumen, el desarrollo y la complejidad creciente de los ataques de XSS persistente destacan la necesidad continua de innovación y vigilancia en la seguridad de las aplicaciones web. La adaptación rápida a nuevas tecnologías y la implementación proactiva de medidas de seguridad son fundamentales para proteger los datos y la privacidad de los usuarios en el entorno digital actual.

3.1.3. Ejemplos destacados de incidentes

3.1.3.1. Casos significativos de XSS persistente en la industria y servicios en línea

Los ataques de Cross-Site Scripting (XSS) persistente han afectado a múltiples industrias y servicios en línea, comprometiendo la integridad y confidencialidad de los datos, y causando daños reputacionales y financieros a las organizaciones. A continuación, se destacan algunos incidentes relevantes:

3.1.3.1.1. Caso 1: MySpace (2005)

En 2005, Samy Kamkar lanzó un gusano XSS en MySpace que se propagaba al visitar perfiles infectados, logrando en un día comprometer más de un millón de cuentas y forzando a MySpace a cerrar temporalmente su sitio (Lane & Clara, 2007). Este incidente mostró la rápida capacidad de propagación y el alto impacto del XSS persistente en redes sociales.

3.1.3.1.2. Caso 2: Yahoo! Mail (2013)

Una vulnerabilidad en Yahoo! Mail permitió a los atacantes robar cookies de sesión y credenciales mediante scripts insertados en su interfaz de correo. Esto comprometió miles de cuentas de usuarios y obligó a Yahoo! a reforzar su seguridad (Trautman & Ormerod, s. f.).

3.1.3.1.3. Caso 3: eBay (2014)

En 2014, un fallo en eBay permitió la ejecución de scripts maliciosos en páginas de productos, exponiendo la información sensible de los usuarios. La respuesta rápida de eBay destacó la importancia de la seguridad en el comercio electrónico (Noori et al., s. f.).

3.1.3.1.4. Caso 4: PayPal (2015)

PayPal sufrió un ataque de XSS persistente que permitió a los atacantes inyectar scripts en notificaciones de pago, con potencial para robar credenciales y realizar transacciones no autorizadas. La vulnerabilidad fue corregida, subrayando la importancia del monitoreo continuo en servicios financieros (McAfee, 2023).

3.1.3.1.5. Caso 5: GitHub (2018)

En GitHub, una vulnerabilidad permitió la inyección de scripts en comentarios de issues y pull requests, lo que facilitó acciones no autorizadas en cuentas de usuarios. GitHub solucionó el problema rápidamente, destacando la colaboración de la comunidad en la identificación de amenazas (Paganini, 2018).

Aunque los casos más notorios de XSS persistente ocurrieron hace algunos años, esta vulnerabilidad sigue siendo relevante. La constante evolución de tecnologías web, como los frameworks modernos y aplicaciones de una sola página (SPA), ofrece nuevos vectores de ataque que los atacantes explotan. Además, el crecimiento de plataformas en línea con grandes volúmenes de datos personales eleva el riesgo de ataques. No todos los incidentes de XSS se publican, ya que muchas empresas los solucionan sin divulgar detalles. Así, el XSS persistente permanece como una amenaza activa en ciberseguridad, y su prevención es fundamental en las prácticas de desarrollo seguro.

4. MÉTODOS DE EXPLOTACIÓN Y VECTORES DE ATAQUE

4.1. Técnicas comunes de explotación

4.1.1. Análisis detallado de cómo los atacantes aprovechan el XSS persistente

Los atacantes explotan la vulnerabilidad del XSS persistente mediante diversas técnicas sofisticadas para inyectar scripts maliciosos en aplicaciones web. Estas técnicas se aprovechan principalmente de la falta de validación rigurosa y sanitización adecuada de los datos ingresados por los usuarios. A continuación, se presenta un análisis de las técnicas más comunes utilizadas por los ciberdelincuentes para llevar a cabo estos ataques:

4.1.1.1. Inyección de Scripts en Formatos de Entrada:

Los atacantes suelen explotar puntos de entrada en aplicaciones web donde los datos de los usuarios no son validados adecuadamente. Estos puntos incluyen formularios de comentarios, campos de búsqueda, perfiles de usuario y áreas de retroalimentación de productos (Shi & Liu, 2022). La falta de validación rigurosa permite que los atacantes inserten scripts maliciosos que quedan almacenados en el servidor, activándose cada vez que un usuario accede a la página comprometida sin percatarse del riesgo.

Este tipo de ataque, conocido como Cross-Site Scripting persistente (XSS persistente), permite a los atacantes robar información de sesión, capturar datos personales, redirigir a sitios maliciosos y manipular acciones del usuario. La gravedad radica en que el código malicioso afecta a todos los usuarios que visitan la página comprometida.

Para mitigar estos ataques, es fundamental aplicar medidas de seguridad en el desarrollo web, como la validación y sanitización de datos, el uso de mecanismos de escape de caracteres especiales, políticas de contenido seguro (CSP) y revisiones constantes de código. Estas prácticas ayudan a proteger a los usuarios y mantener la integridad de las aplicaciones web.

4.1.1.2. Explotación de Funcionalidades de Rich Text Editors:

Muchas aplicaciones web permiten a los usuarios formatear texto utilizando editores enriquecidos, conocidos como rich text editors. Estos editores proporcionan una experiencia de usuario mejorada al permitir la inclusión de elementos de formato como negritas, cursivas, listas, enlaces, e incluso imágenes y videos (Liu et al., 2024). Sin embargo, esta funcionalidad también introduce un riesgo significativo de seguridad si no se gestiona adecuadamente.

Si estos editores enriquecidos no están configurados correctamente para filtrar etiquetas y atributos peligrosos, los atacantes pueden explotar esta vulnerabilidad para insertar scripts maliciosos. Estos scripts pueden estar contenidos dentro del contenido formateado, como en etiquetas de HTML aparentemente inofensivas. Cuando estos scripts no son filtrados y se almacenan junto con el contenido ingresado por el usuario, pueden ser ejecutados automáticamente cada vez que un usuario visualiza la página comprometida.

El impacto de esta vulnerabilidad es considerable, ya que los scripts maliciosos pueden realizar una variedad de acciones dañinas. Entre las posibles consecuencias se incluyen el robo de cookies de sesión, lo que permite a los atacantes suplantar la identidad de los usuarios afectados; la redirección de los usuarios a sitios web fraudulentos; la ejecución de acciones en nombre del usuario sin su consentimiento; y la exposición de información sensible almacenada en la aplicación web. Para mitigar este riesgo, es crucial que los desarrolladores configuren correctamente los editores enriquecidos para que filtren y desinfecten todas las entradas de los usuarios. Esto implica la implementación de listas blancas de etiquetas y atributos permitidos, y la eliminación o codificación de cualquier contenido que pueda ejecutar código. Además, la aplicación de políticas de seguridad de contenido (Content Security Policy, CSP) puede ayudar a limitar la ejecución de scripts no autorizados.

La vigilancia continua y las pruebas de seguridad son esenciales para asegurar que los editores enriquecidos no se conviertan en un vector de ataque. La educación y la concienciación sobre estas vulnerabilidades también son vitales para todos los miembros del equipo de desarrollo, para garantizar que se implementen las mejores prácticas de seguridad desde el inicio del desarrollo hasta la implementación y el mantenimiento de la aplicación web.

4.1.1.3. Manipulación de URL y Parámetros GET:

Los atacantes también pueden aprovechar la falta de sanitización de los parámetros URL y las consultas GET en las aplicaciones web. Estos parámetros son comúnmente utilizados para transmitir datos entre diferentes partes de una aplicación, pero si no se validan adecuadamente, representan un riesgo significativo de seguridad.

Al incluir scripts maliciosos en estos parámetros, los atacantes pueden forzar la ejecución de código malicioso cuando las páginas correspondientes son cargadas por otros usuarios. Este tipo de ataque se conoce como inyección de código en URL o inyección de código GET (Sakib et al., 2024). Los scripts maliciosos pueden ser insertados directamente en los parámetros de la URL, como valores de consulta o fragmentos, o incluso como parte de los nombres de los archivos en las rutas de URL.

Cuando un usuario legítimo accede a una URL comprometida que contiene estos parámetros manipulados, el código malicioso puede ejecutarse en el contexto del navegador del usuario. Esto puede resultar en diversas acciones maliciosas, como robo de cookies de sesión, redirección a sitios web fraudulentos, modificación del contenido de la página, o incluso la ejecución de acciones no autorizadas en nombre del usuario.

Para mitigar este riesgo, es fundamental que las aplicaciones web implementen mecanismos de sanitización y validación estrictos para todos los datos recibidos a través de parámetros URL y consultas GET. Esto incluye validar los tipos de datos esperados, filtrar y escapar caracteres especiales que podrían ser interpretados como código ejecutable, y limitar las acciones que pueden ser realizadas utilizando estos parámetros.

Además, es recomendable utilizar políticas de seguridad de contenido (Content Security Policy, CSP) para restringir la ejecución de scripts no autorizados y garantizar que solo se carguen recursos de fuentes confiables y seguras. La educación continua del equipo de desarrollo sobre las mejores prácticas de seguridad también desempeña un papel crucial en la prevención de este tipo de vulnerabilidades desde las primeras etapas del desarrollo de la aplicación hasta su implementación y mantenimiento.

4.1.1.4. Abuso de Funcionalidades de Carga de Archivos

Las aplicaciones que permiten la carga de archivos son vulnerables a explotaciones si no implementan medidas de seguridad adecuadas. Los atacantes pueden aprovechar esta funcionalidad para cargar archivos que contienen scripts maliciosos, los cuales son ejecutados cuando otros usuarios descargan o visualizan dichos archivos dentro de la aplicación.

Este tipo de ataque se conoce como carga de archivos maliciosos o "file upload attacks". Los scripts maliciosos pueden estar incrustados en diversos tipos de archivos, como imágenes, documentos PDF, hojas de cálculo u otros formatos que permitan la ejecución de código(Johnson, 2008). Cuando un usuario legítimo interactúa con estos archivos dentro de la aplicación comprometida, los scripts maliciosos se activan, lo que puede conducir a acciones dañinas como el robo de cookies de sesión, la ejecución de código arbitrario, o la manipulación y filtración de datos sensibles.

Para mitigar este riesgo, es crucial que las aplicaciones que permiten la carga de archivos implementen las siguientes medidas de seguridad:

- Validación de archivos: Verificar que los archivos cargados cumplen con los tipos y formatos esperados, y que no contienen código o scripts maliciosos incrustados
- Escaneo de archivos: Utilizar herramientas de análisis y escaneo de malware para detectar la presencia de contenido malicioso dentro de los archivos cargados
- Limitación de tipos de archivo: Restringir los tipos de archivos que los usuarios pueden cargar según las necesidades específicas de la aplicación. Por ejemplo, permitir solo imágenes o documentos PDF y bloquear ejecutables o scripts.
- Renombrado de archivos: Cambiar el nombre de los archivos cargados para evitar que puedan ser

ejecutados directamente desde su ubicación en el servidor.

- Aislamiento de archivos: Almacenar los archivos cargados en ubicaciones del servidor que no sean accesibles directamente desde el navegador o desde partes críticas de la aplicación.
- Educación y concienciación: Capacitar a los usuarios sobre los riesgos asociados con la carga y descarga de archivos, y fomentar prácticas seguras como la apertura de archivos solo de fuentes confiables y la descarga de archivos solo cuando sea necesario.

4.1.1.5. Inserción de Scripts en Bases de Datos:

Los atacantes pueden aprovechar la falta de sanitización de datos en campos de texto almacenados en bases de datos, como descripciones de productos, reseñas de usuarios, entradas de blogs u otros contenidos generados por usuarios. Inyectar scripts maliciosos en estos campos es una técnica común utilizada en ataques conocidos como inyección de scripts en base de datos o inyección de scripts almacenados.

Cuando los datos contaminados con scripts maliciosos son almacenados en la base de datos sin un adecuado proceso de sanitización, estos scripts pueden ser recuperados y ejecutados cuando otros usuarios visualizan las páginas correspondientes (Bhattacharyya & de Durgapada, 2024). Este tipo de ataque es particularmente peligroso porque los scripts maliciosos se ejecutan en el contexto del navegador del usuario legítimo, lo que puede resultar en acciones no autorizadas como el robo de cookies de sesión, redirecciones a sitios web fraudulentos, modificación del contenido de la página, o la ejecución de operaciones no deseadas en nombre del usuario.

Para mitigar este riesgo, es esencial implementar prácticas de desarrollo seguro, incluyendo la sanitización de datos mediante filtros y funciones de escapado adecuadas para eliminar o neutralizar cualquier código HTML, JavaScript u otros scripts incrustados en los datos ingresados por los usuarios antes de almacenarlos en la base de datos. Además, es crucial validar y limitar los tipos de datos que los usuarios pueden ingresar en campos de texto, aplicar técnicas de escapado de caracteres especiales al mostrar datos recuperados, implementar políticas de seguridad de contenido (CSP) para restringir la ejecución de scripts no autorizados, y realizar auditorías periódicas de seguridad para identificar y corregir posibles vulnerabilidades.

4.2. Vectores de ataque emergentes

4.2.1. Nuevas tendencias y métodos utilizados por los ciberdelincuentes

En el ámbito de la ciberseguridad, los ciberdelincuentes están constantemente adoptando nuevas estrategias para aprovechar vulnerabilidades como el Cross-Site Scripting (XSS) persistente, adaptándose rápidamente a las medidas de seguridad implementadas. Algunas de las tendencias y métodos emergentes incluyen:

4.2.1.1. Polimorfismo de Scripts:

Los atacantes modifican constantemente el código malicioso utilizando técnicas de ofuscación y polimorfismo para evitar la detección por parte de los sistemas de defensa tradicionales. Esto les permite alterar dinámicamente el código malicioso cada vez que se ejecuta para evitar ser bloqueados por firmas de seguridad estáticas.

4.2.1.2. Uso de APIs y Tecnologías Emergentes:

Con el crecimiento de las aplicaciones web basadas en API (Interfaces de Programación de Aplicaciones)(Carbonell et al., 2018), los atacantes también están aprovechando vulnerabilidades en las APIs mal configuradas o no autenticadas para insertar y ejecutar código malicioso.

4.2.1.3. Ingeniería Social Avanzada:

Además de la explotación técnica, los ciberdelincuentes están refinando sus técnicas de ingeniería social para persuadir a los usuarios para que ejecuten acciones involuntarias que podrían comprometer la seguridad de las aplicaciones web. Esto incluye tácticas como la suplantación de identidad (phishing) y la ingeniería social dirigida a obtener información confidencial o acceso no autorizado.

4.2.1.4. Ataques Dirigidos y Personalizados:

En lugar de ataques genéricos a gran escala, los ciberdelincuentes están optando por métodos más dirigidos y personalizados, adaptados específicamente a las vulnerabilidades conocidas de aplicaciones y usuarios específicos(Xu et al., 2023). Esto aumenta la efectividad de los ataques al reducir la detección y maximizar el impacto.

4.2.1.5. Explotación de Nuevas Superficies de Ataque:

Con la proliferación de dispositivos IoT (Internet de las cosas) y la convergencia de tecnologías, los ciberdelincuentes están explorando nuevas superficies de ataque, como dispositivos domésticos inteligentes y sistemas integrados que pueden tener vulnerabilidades de seguridad no tradicionales, pero igualmente explotables.

Para mitigar estas amenazas emergentes, es crucial que las organizaciones adopten enfoques proactivos y multifacéticos hacia la seguridad cibernética. Esto incluye la implementación de prácticas de desarrollo seguro, la educación continua sobre concienciación en seguridad para empleados y usuarios finales, y la adopción de soluciones de seguridad avanzadas que puedan detectar y responder rápidamente a las nuevas tácticas y técnicas utilizadas por los ciberdelincuentes.

En este contexto, a continuación, se presenta una tabla que muestra la distribución de diferentes tipos de ataques de XSS persistente detectados, seguida de una gráfica que ofrece una representación visual de estos datos. Luego, se proporciona una interpretación detallada de cada categoría de ataque, destacando las tendencias y métodos utilizados por los ciberdelincuentes en el panorama actual de ciberseguridad.

4.2.1.5.1. Análisis de datos de ataques detectados:

Los datos recopilados revelan una distribución significativa de diferentes tipos de ataques, con un enfoque particular en categorías como DDoS, Malware, y IoC Detected(*Cyber Security Attacks*, s. f.-a). Estos datos no solo ilustran la frecuencia de los ataques detectados, sino también su impacto potencial en la seguridad de las aplicaciones web modernas.

ETIQUETAS DE FILA	SUMA DE PUNTAJES DE ANOMALÍA
CONTROL	**1017599,51**
DNS	**341761,34**
IOC DETECTED	167525,85
DDOS	56475,35
INTRUSION	54388,37
MALWARE	56662,13
(EN BLANCO)	174235,49
DDOS	57683,69
INTRUSION	58111,75
MALWARE	58440,05
FTP	**335091,79**
IOC DETECTED	167187,42
DDOS	55579,54
INTRUSION	56190,58
MALWARE	55417,3
(EN BLANCO)	167904,37
DDOS	54171,95
INTRUSION	57503,81
MALWARE	56228,61
HTTP	**340746,38**
IOC DETECTED	172156,41

DDOS	56828,19
INTRUSION	56931,37
MALWARE	58396,85
(EN BLANCO)	168589,97
DDOS	56816,17
INTRUSION	54135,25
MALWARE	57638,55
DATA	**986939,42**
DNS	**331906,64**
IOC DETECTED	165144,73
DDOS	55628,2
INTRUSION	55348,11
MALWARE	54168,42
(EN BLANCO)	166761,91
DDOS	57878,8
INTRUSION	54181,03
MALWARE	54702,08
FTP	**322825,9**
IOC DETECTED	163919,02
DDOS	52754,85
INTRUSION	56533,54
MALWARE	54630,63
(EN BLANCO)	158906,88
DDOS	55700,64
INTRUSION	51209,54
MALWARE	51996,7
HTTP	**332206,88**
IOC DETECTED	166217,84
DDOS	57930,77
INTRUSION	53603,87
MALWARE	54683,2
(EN BLANCO)	165989,04
DDOS	57108,28
INTRUSION	54821,34
MALWARE	54059,42
TOTAL GENERAL	**2004538,93**

Tabla 1: Sumas de Puntajes de Anomalía por Categoría de Ataque (Cyber Security Attacks, s. f.-b)

La tabla proporciona una visión detallada de cómo los ataques de XSS persistente están distribuidos en diferentes categorías, resaltando la prevalencia de incidentes como DDoS y Malware. Estos resultados subrayan la importancia de implementar estrategias robustas de mitigación y utilizar herramientas avanzadas de detección, como OWASP ZAP y Burp Suite, para proteger eficazmente las aplicaciones web contra estas amenazas emergentes.

4.2.1.5.2. Gráfica de distribución de ataques:

La gráfica asociada muestra visualmente la distribución porcentual de cada tipo de ataque, proporcionando una representación clara y concisa de las tendencias observadas en los datos analizados.

Ilustración 2: Distribución de Puntajes de Anomalía por Tipo de Servicio y Tipo de Amenaza (*Cyber Security Attacks*, s. f.-b)

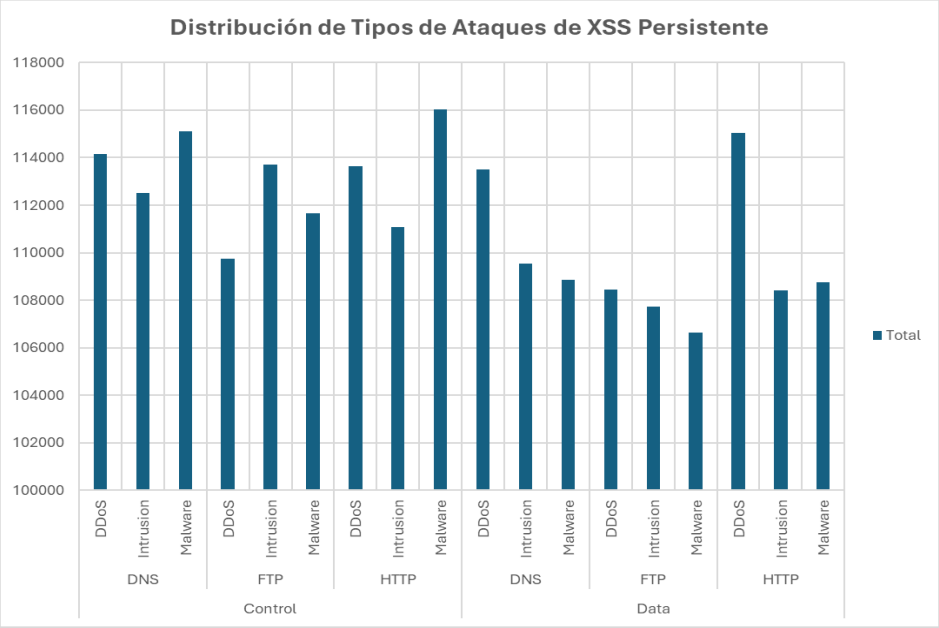
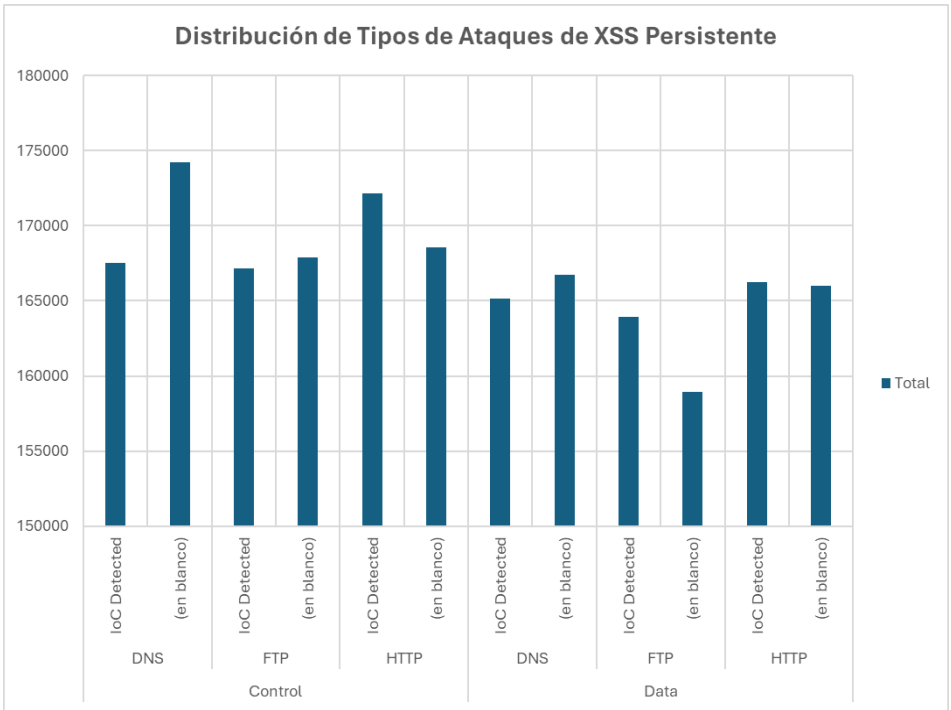


Ilustración 3 Puntajes de Anomalía por Tipo de Servicio y Detección de Indicadores de Compromiso (IoC)(*Cyber Security Attacks*, s. f.-b)



La gráfica proporciona una visión detallada de la distribución de diferentes tipos de ataques de XSS persistente detectados en una muestra representativa de datos cibernéticos. Observamos que los tipos de ataques más frecuentes incluyen DDoS, Intrusiones y Malware, que representan una parte significativa del panorama de amenazas actuales. Esto sugiere que los ciberdelincuentes están diversificando sus métodos para aprovechar vulnerabilidades de XSS persistente, utilizando desde ataques de denegación de servicio distribuido (DDoS) hasta intrusiones dirigidas y distribución de malware a través de aplicaciones web vulnerables.

La distribución muestra que los ataques de DDoS y Malware son especialmente prominentes, lo cual es consistente con las tendencias actuales donde los ciberdelincuentes buscan no solo interrumpir servicios con DDoS, sino también comprometer sistemas para la distribución de malware, potencialmente comprometiendo la integridad y confidencialidad

de los datos de los usuarios. Esta concentración en ciertos tipos de ataques destaca la importancia de implementar medidas de seguridad robustas y continuamente actualizadas para mitigar estos riesgos.

Además, la gráfica resalta la presencia de técnicas emergentes como la ingeniería social avanzada y el polimorfismo de scripts, indicando que los atacantes están adaptando y refinando sus estrategias para evadir las defensas tradicionales. Esta adaptación subraya la necesidad de una defensa cibernética dinámica que no solo se enfoque en las vulnerabilidades conocidas, sino que también esté preparada para enfrentar nuevas tácticas y métodos de ataque.

5. MEDIDAS DE MITIGACIÓN Y BUENAS PRÁCTICAS

5.1. Estrategias de prevención y mitigación

La prevención y mitigación del XSS persistente requieren un enfoque integral que combine técnicas de desarrollo seguro, políticas de seguridad y herramientas especializadas. A continuación, se presentan métodos efectivos para protegerse contra esta vulnerabilidad:

5.1.1. Validación y Sanitización de Datos:

Validación de Entrada: Implementar validaciones estrictas en todos los puntos de entrada de datos, asegurándose deque solo se permitan caracteres y formatos esperados **(**OWASP**,** 2023**)**. Utilizar listas blancas para definir los caracteres aceptables, una recomendación ampliamente aceptada en ciberseguridad **(**Arziaev**,** 2024**).**

Sanitización de Datos: Sanitizar los datos de entrada y salida para eliminar contenido malicioso antes de almacenar lo en el servidor o mostrarlo en el navegador (OWASP, 2023). Las funciones de sanitización deben aplicarse en todos los vectores de entrada posibles.

5.1.2. Codificación de Salida:

HTML Encoding: Codificar los caracteres especiales antes de enviarlos al navegador para evitar que se interpreten como código HTML (A. Gupta, 2023). Esto incluye caracteres como <, >, &, ", y '.

JavaScript Encoding: Codificar datos que se insertan en contextos JavaScript para prevenir la inyección de scripts maliciosos(OWASP, 2023).

5.1.3. Uso de Controles de Seguridad en el Navegador:

Content Security Policy (CSP): Implementar una política de seguridad de contenido (CSP) para restringir las fuentes de scripts permitidas y prevenir la ejecución de scripts no autorizados (Xia et al., 2024). Una política bien configurada puede reducir significativamente el riesgo de XSS.

HTTP Headers: Configurar encabezados HTTP de seguridad como X-XSS-Protection y X-Content-Type-Options para fortalecer la protección contra ataques de XSS (Mozilla Fundation, s. f.).

5.1.4. Revisiones y Auditorías de Código:

Revisiones de Código: Realizar revisiones de código regulares para identificar y corregir posibles vulnerabilidades de XSS persistente (Schoenfeld, 2024). Utilizar herramientas automatizadas para detectar patrones comunes de vulnerabilidades y asegurar que todos los datos sean correctamente validados y sanitizados.

Auditorías de Seguridad: Llevar a cabo auditorías de seguridad periódicas, tanto internas como externas, para evaluar la efectividad de las medidas de seguridad implementadas y descubrir posibles fallos (OWASP, 2023).

5.1.5. Implementación de Frameworks y Librerías Seguras:

Frameworks Seguros: Utilizar frameworks de desarrollo web que ofrezcan protección integrada contra XSS, como Angular, React y Vue.js, que incluyen mecanismos de escape automático de datos(Ahmad et al., 2024). Estos frameworks pueden ayudar a prevenir errores comunes que pueden conducir a vulnerabilidades de XSS.

Actualizaciones y Parches: Mantener todas las bibliotecas y frameworks actualizados con los últimos parches de seguridad para mitigar vulnerabilidades conocidas(OWASP, 2023).

5.1.6. Educación y Capacitación:

Capacitación de Desarrolladores: Capacitar a los desarrolladores sobre las mejores prácticas de seguridad y la importancia de la validación y sanitización de datos(Oliveira et al., 2022). La educación continua ayuda a mantener al equipo actualizado sobre las últimas amenazas y técnicas de mitigación.

Conciencia de Seguridad: Fomentar una cultura de seguridad dentro de la organización, donde todos los miembros del equipo comprendan los riesgos asociados con el XSS persistente y la importancia de las medidas preventivas (Microsft Security, s. f.).

5.1.7. Pruebas de Penetración y Escaneo de Vulnerabilidades:

Pruebas de Penetración: Realizar pruebas de penetración para identificar y explotar posibles vulnerabilidades en las aplicaciones web (Budyal et al., 2024). Esto ayuda a descubrir y mitigar fallos antes de que los atacantes puedan explotarlos.

Escaneo de Vulnerabilidades: Utilizar herramientas de escaneo de vulnerabilidades para detectar automáticamente posibles puntos de entrada para ataques de XSS persistente (OWASP, 2023).

5.1.8. Gestión de Sesiones y Autenticación:

Gestión de Sesiones: Implementar técnicas de gestión segura de sesiones para minimizar el riesgo de robo de sesiones a través de XSS (Jain et al., 2023). Esto incluye el uso de tokens de sesión seguros y la expiración automática de sesiones inactivas.

Autenticación Multifactor (MFA): Reforzar la autenticación de usuarios mediante la implementación de autenticación multifactor para reducir el impacto de posibles compromisos de cuentas.

5.1.9. Implementación de Prácticas de Seguridad en el Ciclo de Vida del Software

La incorporación de prácticas de seguridad en el ciclo de vida del software es esencial para prevenir vulnerabilidades como XSS persistente. Los principios de diseño seguro, incluidos en documentos como la guía OWASP SDLC (OWASP, 2023), ayudan a integrar seguridad en cada etapa del desarrollo.

Planificación y Análisis de Requisitos: Análisis de Riesgos: Realizar un análisis de riesgos para identificar posibles vulnerabilidades en fases tempranas (OWASP, 2023).

Requisitos de Seguridad: Definir requisitos de validación y sanitización de datos desde el inicio.

Diseño:

Principios de Seguridad desde el Diseño: Diseñar sistemas con mecanismos de defensa en profundidad y privilegios mínimos, siguiendo recomendaciones de diseño seguro (Computer Security Division, 2023).

Modelado de Amenazas: Realizar modelados de amenazas para identificar vectores de ataque de XSS persistente.

Desarrollo:

Buenas Prácticas de Codificación: Utilizar patrones de codificación segura y evitar funciones inseguras.

Librerías y Frameworks Seguros: Utilizar librerías y frameworks actualizados con protección contra XSS.

Pruebas:

Pruebas de Seguridad: Incluir pruebas de seguridad en el proceso de integración continua.

Pruebas de Usuario: Asegurar que la usabilidad de la aplicación no se vea afectada.

Implementación:

Configuración Segura del Entorno: Aplicar configuraciones seguras en producción, siguiendo guías de seguridad (OWASP, 2023).

Despliegue Seguro: Asegurar que las dependencias de terceros estén actualizadas.

Mantenimiento:

Monitoreo y Detección: Monitoreo de actividades sospechosas con soluciones como IDS.

Gestión de Parches y Actualizaciones: Mantener software y componentes actualizados con los últimos parches de seguridad.

6. HERRAMIENTAS Y TECNOLOGÍAS DE DETECCIÓN

La detección y mitigación de vulnerabilidades de Cross-Site Scripting (XSS) persistente requiere el uso de herramientas especializadas y tecnologías avanzadas. A continuación, se presenta una revisión de algunas de las herramientas y servicios más efectivos disponibles para identificar y gestionar este tipo de vulnerabilidades.

6.1. Herramientas Disponibles para la Detección de XSS Persistente

6.1.1. OWASP ZAP (Zed Attack Proxy):

OWASP ZAP es una herramienta gratuita y de código abierto diseñada para encontrar vulnerabilidades en aplicaciones web (Viegas & Kuyucu, 2022). Es altamente eficaz para detectar XSS persistente y otras vulnerabilidades de seguridad comunes. Proporciona una variedad de funciones, incluyendo:

Escaneo Activo y Pasivo: Detecta vulnerabilidades a través de escaneos automatizados.

Proxy Intercept: Permite la inspección y modificación del tráfico HTTP/HTTPS entre el navegador y la aplicación web.

Fuzzing: Prueba la aplicación con datos inesperados o aleatorios para identificar fallos de seguridad.

6.1.2. Burp Suite:

Burp Suite es una plataforma integrada utilizada para realizar pruebas de seguridad en aplicaciones web (Rahalkar, 2021). Desarrollado por PortSwigger, es una de las herramientas más utilizadas por profesionales de la seguridad. Ofrece:

Escáner Automático: Identifica automáticamente vulnerabilidades de XSS persistente.

Intercepción de Proxies: Inspecciona y modifica el tráfico HTTP/HTTPS.

Intruder y Repeater: Herramientas para automatizar ataques y realizar pruebas manuales.

6.1.3. Acunetix:

Acunetix es una solución comercial que ofrece capacidades avanzadas de escaneo de seguridad web (Aljebry et al., 2022). Es particularmente eficaz en la detección de XSS persistente. Sus características incluyen:

Escaneo Automático: Detecta una amplia gama de vulnerabilidades de seguridad.

Desinfección de Resultados Falsos Positivos: Minimiza los falsos positivos para un análisis más preciso.

Integración CI/CD: Se integra con sistemas de desarrollo continuo para asegurar la seguridad en todo el ciclo de vida del software.

6.1.4. Netsparker:

Netsparker es otra herramienta comercial reconocida por su capacidad para detectar y explotar automáticamente vulnerabilidades de XSS persistente (Cvitić et al., 2022). Sus características destacadas incluyen:

Escaneo Preciso: Alta precisión en la detección de vulnerabilidades sin falsos positivos.

Automatización: Automatiza la identificación y explotación de vulnerabilidades.

Integraciones: Se integra con múltiples plataformas CI/CD y herramientas de desarrollo.

6.1.5. Arachni:

Arachni es una herramienta de escaneo de seguridad web de código abierto que se destaca por su capacidad para detectar XSS persistente(Rajić et al., 2023). Ofrece:

Escaneo Distribuido: Permite realizar escaneos distribuidos en múltiples instancias.

Compatibilidad con Plugins: Extensible con plugins adicionales para aumentar sus capacidades.

Reporting Avanzado: Genera informes detallados sobre las vulnerabilidades detectadas.

6.1.6. W3AF (Web Application Attack and Audit Framework):

W3AF es una plataforma de código abierto para encontrar y explotar vulnerabilidades en aplicaciones web(Ramos Cruz et al., 2024). Es altamente modular y ofrece:

Módulos de Ataque y Auditoría: Módulos específicos para detectar y explotar XSS persistente.

Extensibilidad: Facilita la adición de nuevos módulos para detectar diversas vulnerabilidades.

Interfaz de Línea de Comandos y Gráfica: Ofrece tanto una interfaz CLI como GUI.

6.2. Evaluación Comparativa de Herramientas

La selección de herramientas para la detección y mitigación de XSS persistente depende de diversos factores como el entorno de desarrollo, las necesidades específicas del proyecto, el presupuesto y la experiencia del equipo de seguridad. A continuación, se presenta una evaluación comparativa de algunas de las herramientas mencionadas previamente, junto con recomendaciones para su selección y uso en diferentes contextos.

6.2.1. Comparativa de Herramientas

Herramienta	Tipo	Características Principales	Ventajas	Limitaciones
OWASP ZAP	Código Abierto	Escaneo activo/pasivo, Proxy intercept, Fuzzing	Gratuito, amplio soporte comunitario	Curva de aprendizaje para usuarios novatos
Burp Suite	Comercial	Escáner automático, Intercepción de proxies, Intruder y Repeater	Potente y versátil, utilizado por profesionales	Costoso, especialmente la versión Pro
Acunetix	Comercial	Escaneo automático, Desinfección de falsos positivos, CI/CD	Alta precisión, integración CI/CD	Costoso, puede ser excesivo para proyectos pequeños
Netsparker	Comercial	Escaneo preciso, Automatización, Integraciones	Alta precisión sin falsos positivos, automatización	Costoso, especialmente para pequeñas empresas
Arachni	Código Abierto	Escaneo distribuido, Compatibilidad con plugins, Reporting	Gratuito, extensible	Requiere conocimientos técnicos avanzados
W3AF	Código Abierto	Módulos de ataque y auditoría, Extensibilidad, CLI y GUI	Gratuito, altamente extensible	Interfaz menos intuitiva que herramientas comerciales
Veracode	Servicio en	Análisis estático y dinámico, Integración DevOps,	Solución integral, soporte	Suscripción costosa

	la nube	Capacitación	y capacitación	
Checkmarx	Comercial	Análisis profundo, Integración DevSecOps, Dashboard	Excelente integración con entornos DevOps	Costoso, requiere capacitación
Qualys WAS	Servicio en la nube	Escaneo automático, Remediación, Conformidad	Fácil de usar, informes detallados	Costoso para pequeñas organizaciones
AppSpider	Comercial	Escaneo dinámico, Cobertura completa, Reporting	Excelente cobertura de aplicaciones modernas	Costoso, requiere configuración inicial

Ilustración 4 Comparación de Herramientas de Detección de XSS Persistente (Elaboración Propia)

6.2.2. Recomendaciones para la Selección y Uso de Herramientas

6.2.2.1. Proyectos Pequeños y Presupuestos Limitados:

OWASP ZAP: Ideal para proyectos con recursos limitados debido a su naturaleza gratuita y su amplio soporte comunitario. Recomendado para pequeñas empresas y desarrolladores individuales.

Arachni y W3AF: Ambas herramientas de código abierto ofrecen capacidades avanzadas sin costo, aunque requieren conocimientos técnicos más profundos.

6.2.2.2. Equipos Profesionales y Empresas Medianas:

Burp Suite (Community y Pro): La versión gratuita (Community) es adecuada para aprender y realizar pruebas básicas, mientras que la versión Pro ofrece funciones avanzadas necesarias para equipos profesionales.

Netsparker: Ideal para empresas medianas que requieren una herramienta con alta precisión y automatización. Su costo se justifica por la reducción en falsos positivos y la eficiencia en la detección.

6.2.2.3. Grandes Empresas y Entornos Corporativos:

Acunetix y Qualys WAS: Estas soluciones son adecuadas para grandes empresas debido a su capacidad para integrarse con CI/CD y ofrecer soporte detallado. A pesar de su costo, proporcionan un valor significativo en entornos corporativos.

Checkmarx y Veracode: Recomendados para organizaciones que buscan una solución integral que incluya análisis de seguridad en todas las fases del ciclo de vida del software. Ofrecen soporte, capacitación y capacidades avanzadas de integración.

6.2.2.4. Proyectos con Enfoque en DevOps y CI/CD:

Veracode y Checkmarx: Ambas herramientas se integran perfectamente en entornos DevOps y CI/CD, permitiendo a los equipos de desarrollo mantener la seguridad como una parte integral del proceso de desarrollo.

6.2.2.5. Educación y Capacitación:

OWASP ZAP y Burp Suite Community: Excelentes herramientas para educar y capacitar a nuevos profesionales en el campo de la seguridad de aplicaciones web debido a su accesibilidad y funcionalidad robusta.

7. ANÁLISIS DE RESULTADOS

7.1. Estado Actual de XSS Persistente

En la revisión sistemática de la literatura se encontró que las vulnerabilidades de XSS persistente continúan siendo un problema significativo en el desarrollo de aplicaciones web. A pesar de los avances en las prácticas de seguridad, la insuficiente validación de entradas y la dependencia en contenido dinámico siguen siendo las principales causas de

estas vulnerabilidades.

7.2. Medidas de Mitigación

Las estrategias más efectivas para mitigar el XSS persistente incluyen la validación y sanitización rigurosa de datos de entrada, la codificación adecuada de la salida, y la implementación de controles de seguridad en el navegador. La educación continua y la capacitación de desarrolladores también son esenciales para mantener prácticas de codificación seguras.

7.3. Herramientas de Detección y Mitigación

Se evaluaron diversas herramientas de detección de XSS persistente:

OWASP ZAP (Zed Attack Proxy): Una herramienta de código abierto eficaz para identificar vulnerabilidades de XSS y otras amenazas en aplicaciones web.

Burp Suite: Una herramienta profesional ampliamente utilizada que ofrece un conjunto completo de funcionalidades para detectar y explotar vulnerabilidades de XSS.

Acunetix: Conocida por su precisión y capacidad de escaneo profundo, esta herramienta es ideal para entornos empresariales.

Netsparker: Ofrece una solución automatizada y precisa para la detección de XSS persistente, con énfasis en la facilidad de uso.

Arachni: Una herramienta de código abierto que proporciona una excelente opción para aquellos con presupuestos limitados.

W3AF (Web Application Attack and Audit Framework): Otra opción de código abierto que es flexible y extensible, ideal para proyectos pequeños y medianos.

7.4. Impactos de XSS Persistente

El impacto de las vulnerabilidades de XSS persistente es considerable, incluyendo:

Riesgos Financieros: Pérdidas monetarias debido a fraudes y ataques.

Daño a la Reputación: La confianza del cliente se ve afectada negativamente tras incidentes de seguridad.

Cumplimiento Normativo: Las organizaciones pueden enfrentar sanciones por incumplimiento de regulaciones como GDPR y CCPA.

Costos de Recuperación: Gastos asociados con la respuesta a incidentes y la reparación de sistemas comprometidos.

7.5. Comparativa de Herramientas

La evaluación comparativa de las herramientas reveló que las soluciones comerciales, como Burp Suite y Acunetix, suelen ser más completas y precisas en comparación con las herramientas de código abierto. Sin embargo, herramientas como OWASP ZAP y Arachni ofrecen alternativas viables para aquellos con limitaciones presupuestarias.

7.6. Recomendaciones

Para proyectos pequeños y presupuestos limitados, las herramientas de código abierto como OWASP ZAP y Arachni son recomendadas. Para equipos profesionales y empresas medianas, herramientas como Netsparker y Burp Suite ofrecen una mayor precisión y un conjunto de funcionalidades más robusto. En grandes empresas y entornos corporativos, Acunetix y Burp Suite son las opciones preferidas debido a su profundidad y capacidad de personalización.

8. DISCUSIÓN

La evaluación de la vulnerabilidad XSS persistente y su contexto actual pone de manifiesto varios puntos clave que deben ser considerados para una comprensión completa y una gestión efectiva de esta amenaza.

8.1. Relevancia Continua de XSS Persistente

A pesar de los avances en la seguridad de las aplicaciones web, XSS persistente sigue siendo una amenaza significativa. La persistencia de estas vulnerabilidades se debe en gran parte a prácticas insuficientes de validación y sanitización de datos, así como a la complejidad creciente de las aplicaciones web modernas que manejan una gran cantidad de contenido dinámico. Los datos revisados confirman que, incluso con la existencia de herramientas avanzadas y técnicas de mitigación, las aplicaciones web continúan siendo susceptibles a estos ataques.

8.2. Impacto de las Medidas de Mitigación

Las medidas de mitigación y las buenas prácticas identificadas en esta revisión, como la validación de datos, la codificación de salida y el uso de políticas de seguridad como Content Security Policy (CSP), han demostrado ser efectivas para reducir el riesgo de XSS persistente. Sin embargo, la implementación completa y rigurosa de estas prácticas no siempre es efectiva debido a diversos factores, como la falta de capacitación y la falta de recursos en las organizaciones. La adopción de estas estrategias debe ser vista como un proceso continuo y adaptativo para mantenerse al día con las nuevas técnicas de ataque y vulnerabilidades emergentes.

8.3. Efectividad de las Herramientas de Detección

La revisión de herramientas de detección muestra que las soluciones comerciales, como Burp Suite y Acunetix, ofrecen una cobertura más amplia y funcionalidades avanzadas que son especialmente útiles en entornos empresariales complejos. Por otro lado, las herramientas de código abierto, como OWASP ZAP y Arachni, proporcionan opciones viables y accesibles, aunque pueden carecer de algunas de las características avanzadas ofrecidas por las soluciones comerciales. La elección de herramientas debe alinearse con el contexto y los requisitos específicos de cada proyecto o organización.

8.4. Tendencias Emergentes en XSS Persistente

Las tendencias emergentes en XSS persistente, como el polimorfismo de scripts y la explotación de nuevas superficies de ataque, destacan la necesidad de una vigilancia continua y una adaptación rápida a las nuevas amenazas. Los ciberdelincuentes están desarrollando técnicas más sofisticadas y personalizadas, lo que requiere que las organizaciones estén siempre actualizadas con las mejores prácticas de seguridad y las últimas herramientas de detección. La incorporación de estas prácticas debe ser parte integral del ciclo de vida del desarrollo de software para garantizar una protección robusta.

8.5. Desafíos en la Implementación de Prácticas de Seguridad

A pesar de las mejores prácticas y las herramientas disponibles, la implementación efectiva de medidas de seguridad enfrenta varios desafíos. Estos incluyen la resistencia al cambio por parte de los desarrolladores, la falta de recursos financieros para herramientas de seguridad avanzadas, y la necesidad de formación continua para mantenerse al día con las amenazas emergentes. Las organizaciones deben abordar estos desafíos mediante una combinación de capacitación, inversión en herramientas adecuadas y la integración de prácticas de seguridad desde las fases iniciales del desarrollo.

8.6. Importancia de la Educación y la Capacitación

La formación continua en seguridad cibernética es crucial para mantener a los desarrolladores informados sobre las últimas vulnerabilidades y técnicas de mitigación. Los programas de capacitación deben incluir información actualizada sobre XSS persistente y otras amenazas emergentes, así como proporcionar habilidades prácticas para la implementación de medidas de seguridad efectivas. La integración de la seguridad en la educación y la capacitación ayuda a crear una cultura de seguridad dentro de las organizaciones, lo que puede reducir significativamente la incidencia de vulnerabilidades.

8.7. Necesidad de Enfoques Proactivos y Multifacéticos

Para abordar eficazmente la vulnerabilidad de XSS persistente, es esencial adoptar un enfoque proactivo y multifacético. Esto incluye la integración de prácticas de seguridad en cada etapa del ciclo de vida del desarrollo, la utilización de herramientas de detección avanzadas y la implementación de estrategias de mitigación basadas en las mejores prácticas. Un enfoque integral no solo ayuda a prevenir la explotación de vulnerabilidades existentes, sino que también prepara a las organizaciones para enfrentar futuras amenazas de manera efectiva.

9. CONCLUSIONES

La investigación presentada en este artículo de revisión sobre Cross-Site Scripting (XSS) persistente ofrece una visión detallada de esta significativa amenaza en la seguridad cibernética. A través de un análisis, se ha abordado desde la definición del XSS persistente hasta su impacto en la integridad y confidencialidad de los datos, contribuyendo a un entendimiento más amplio de esta vulnerabilidad crítica. Se discutió la evolución histórica del XSS persistente, destacando su avance desde ataques simples hasta técnicas de inyección sofisticadas en aplicaciones modernas. Esta evolución ha aumentado la complejidad de los métodos de explotación, con ciberdelincuentes que aprovechan vulnerabilidades técnicas y tácticas de ingeniería social. El estudio resaltó los impactos del XSS persistente, que compromete la privacidad de los usuarios, roba credenciales y facilita acciones maliciosas. Estos riesgos afectan tanto a individuos como a organizaciones, que pueden enfrentar pérdidas financieras, daño reputacional y consecuencias legales.
En cuanto a mitigación, se evaluaron diversas estrategias y prácticas recomendadas, como la validación estricta de datos, codificación segura y el uso de herramientas avanzadas de detección. Se enfatizó la importancia de integrar la seguridad desde las etapas iniciales del ciclo de vida del software hasta el mantenimiento continuo.

En conclusión, este documento ha subrayado la necesidad urgente de una preparación integral y continua en seguridad cibernética. Ninguna organización está exenta de riesgos relacionados con el XSS persistente, y la complacencia en seguridad puede llevar a consecuencias devastadoras. Es crucial que las organizaciones adopten un enfoque proactivo hacia la seguridad, incorporando prácticas sólidas de desarrollo seguro, educación continua y tecnologías avanzadas de defensa. La colaboración entre equipos de desarrollo, operaciones y seguridad es esencial para implementar estrategias de mitigación efectivas y responder rápidamente a incidentes. Finalmente, la protección contra el XSS persistente no es solo una responsabilidad técnica, sino un imperativo estratégico que debe abordarse a nivel organizacional. Esto permitirá a las organizaciones fortalecer su resiliencia frente a amenazas emergentes y proteger sus activos digitales en un entorno cada vez más complejo y dinámico.

10. REFERENCIAS

Ahmad, S., Mehfuz, S., Urooj, S., & Alsubaie, N. (2024). Machine learning-based intelligent security framework for secure cloud key management. *Cluster Computing*, 1-27. https://doi.org/10.1007/s10586-024-04288-8

Ajay Pal Singh, C., & Ashwani Kumar, A. (2022). *Full article: Dynamic Deep Genomics Sequence Encoder for Managed File Transfer*. https://tandfonline.proxyusc.elogim.com/doi/full/10.1080/03772063.2022.2060869

Aljebry, A. F., Alqahtani, Y. M., & Sulaiman, N. (2022). Analyzing Security Testing Tools for Web Applications. En *International Conference on Innovative Computing and Communications* (pp. 411-419). Springer, Singapore. https://doi.org/10.1007/978-981-16-2594-7_34

Arziaev, A. (2024). Data Migration Testing and Validation. En *Data Migration Management for SAP S/4HANA* (pp. 79-91). Apress, Berkeley, CA. https://doi.org/10.1007/979-8-8688-0333-8_5

Bates, D., Barth, A., & Jackson, C. (2010). Regular expressions considered harmful in client-side XSS filters. *Proceedings of the 19th international conference on World wide web*, 91-100. https://doi.org/10.1145/1772690.1772701

Bhattacharyya, A., & de Durgapada, C. (2024). *Graph Database: Re-engineering Methodologies Relational to NOSQL Databases | SpringerLink*. https://springerlink.proxyusc.elogim.com/chapter/10.1007/978-981-97-1260-1_12

Budyal, V., Vaibhav, A. V., Akshay, C. U., Ishika, N., & Unnathi, G. (2024). Cyber-Attack Analysis Using Vulnerability Assessment and Penetration Testing. *Cyber Security and Digital Forensics*, 123-134. https://doi.org/10.1007/978-981-99-9811-1_10

Carbonell, J., Sánchez-Esguevillas, A., & Carro, B. (2018). Easing the assessment of emerging technologies in technology observatories. Findings about patterns of dissemination of emerging technologies on the internet. *Technology Analysis & Strategic Management*, *30*(1), 113-129. https://doi.org/10.1080/09537325.2017.1337886

Computer Security Division, I. T. L. (2023, diciembre 20). *Cybersecurity of Genomic Data: NIST IR 8432 | CSRC*. CSRC | NIST. https://csrc.nist.gov/news/2023/cybersecurity-of-genomic-data-nist-ir-8432

Cvitić, I., Peraković, D., Periša, M., & Sekondo, M. (2022). Exploring the Applicability of Open-Source Tools for Web Application Cybersecurity Improvement. *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, 64-79. https://doi.org/10.1007/978-3-031-15101-9_5

Cyber Security Attacks. (s. f.-a). Recuperado 15 de julio de 2024, de https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks

Cyber Security Attacks. (s. f.-b). Recuperado 29 de octubre de 2024, de https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks

Endler, D. (2002). *Evolution of Cross site Scripting*.

Grossman. (s. f.). Recuperado 29 de octubre de 2024, de https://theswissbay.ch/pdf/Gentoomen%20Library/Security/Cross%20Site%20Scripting%20Attacks%20Xss%20Exploits%20and%20Defense.pdf

Grossman, J. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Syngress.

Gupta, A. (2023). Codes and Coding. En *Qualitative Methods and Data Analysis Using ATLAS.ti* (pp. 99-125). Springer, Cham. https://doi.org/10.1007/978-3-031-49650-9_4

Gupta, S., & Sharma, L. (2012). Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense. *International Journal of Computer Applications (0975 – 8887)*, *60*, 28-33. https://doi.org/10.5120/9762-3594

Heiderich, M., Niemietz, M., Schuster, F., Holz, T., & Schwenk, J. (2012). Scriptless attacks: Stealing the pie without touching the sill. *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 760-771. https://doi.org/10.1145/2382196.2382276

Hydara, I., Sultan, A. B. Md., Zulzalil, H., & Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS) – A systematic literature review. *Information and Software Technology*, *58*, 170-186. https://doi.org/10.1016/j.infsof.2014.07.010

Jain, U., Tripathi, A., Kumar, S., & Kumar, G. (2023). Simple, secure and lightweight authentication protocol with session-key generation for IIoT device in IIoT networks. *Microsystem Technologies*, 1-13. https://doi.org/10.1007/s00542-023-05566-y

Johnson, M. E. (2008). Information Risk of Inadvertent Disclosure: An Analysis of File-Sharing Risk in the Financial Supply Chain. *Journal of Management Information Systems*, *25*(2), 97-124. https://doi.org/10.2753/MIS0742-1222250205

Khan, M. S., Siam, R. S. F., & Adnan, M. A. (2024). A framework for checking and mitigating the security vulnerabilities of cloud service RESTful APIs. *Service Oriented Computing and Applications*, 1-22. https://doi.org/10.1007/s11761-024-00404-z

Kirda, E., Kruegel, C., Vigna, G., & Jovanovic, N. (2006). Noxes: A client-side solution for mitigating cross-site scripting

attacks. *Proceedings of the 2006 ACM Symposium on Applied Computing*, 330-337. https://doi.org/10.1145/1141277.1141357

Kurniawan, A., Kurniawan, A., Soesanto, C., & Wijaya, J. E. C. (2015). CodeR: Real-time Code Editor Application for Collaborative Programming. *Procedia Computer Science*, *59*, 510-519. https://doi.org/10.1016/j.procs.2015.07.531

Lane, B. H., & Clara, S. (2007). *A WhiteHat Security Whitepaper*.

Liu, M., Li, Y., Su, Y., & Li, H. (2024). Text Complexity of Chinese Elementary School Textbooks: Analysis of Text Linguistic Features Using Machine Learning Algorithms. *Scientific Studies of Reading*, *28*(3), 235-255. https://doi.org/10.1080/10888438.2023.2244620

McAfee. (2023, enero 24). The PayPal Breach – Who Was Affected and How You Can Protect Yourself. *McAfee Blog*. https://www.mcafee.com/blogs/security-news/the-paypal-breach-who-was-affected-and-how-you-can-protect-yourself/

Microsft Security. (s. f.). *Informe de protección digital de Microsoft (MDDR) 2023*. Recuperado 29 de octubre de 2024, de https://www.microsoft.com/es-es/security/security-insider/microsoft-digital-defense-report-2023

Mohammadi, M., Chu, B., Richter Lipford, H., & Murphy-Hill, E. (2016). *Pruebas unitarias de seguridad web automáticas | Actas del 11.º Taller internacional sobre automatización de pruebas de software*. https://dl.acm.org/doi/abs/10.1145/2896921.2896929

Mozilla Fundation. (s. f.). *Internet para la gente, no para el lucro*. Mozilla. Recuperado 29 de octubre de 2024, de https://www.mozilla.org/es-ES/

Noori, I., Patabendi, M., & Malik, A. (s. f.). *Cyber attack on ebay—2014*.

Oliveira, D., Assunção, W. K. G., Garcia, A., Fonseca, B., & Ribeiro, M. (2022). Developers' perception matters: Machine learning to detect developer-sensitive smells. *Empirical Software Engineering*, *27*(7), 1-44. https://doi.org/10.1007/s10664-022-10234-2

OWASP. (2023). *Secuencias de comandos entre sitios (XSS) | Fundación OWASP*. https://owasp.org/www-community/attacks/xss/

Paganini, P. (2018, marzo 2). *Github hit by the biggest-ever DDoS attack that peaked 1.35 Tbs*. Security Affairs. https://securityaffairs.com/69762/hacking/github-largest-ddos-attack.html

Pintus Antonio, Davide Carboni, & Andrea Piras. (s. f.). *The anatomy of a large scale social web for internet enabled objects | Proceedings of the Second International Workshop on Web of Things*. Recuperado 29 de octubre de 2024, de https://dl.acm.org/doi/abs/10.1145/1993966.1993975

Rahalkar, S. (2021). *A Complete Guide to Burp Suite*. https://springerlink.proxyusc.elogim.com/book/10.1007/978-1-4842-6402-7

Rajić, B., Stanisavljević, Ž., & Vuletić, P. (2023). Early web application attack detection using network traffic analysis. *International Journal of Information Security*, *22*(1), 77-91. https://doi.org/10.1007/s10207-022-00627-1

Ramos Cruz, B., Andreu-Pérez, J., & Martínez, L. (2024). *Scopus—Detalles del documento—Un enfoque proactivo para evaluar la seguridad de las aplicaciones web a través de la integración de herramientas de seguridad en una plataforma de orquestación de seguridad*. https://doi.org/10.1016/j.cose.2022.102886

Sakib, M. N., Robin, K. H., Siddika, A., & Nahar, S. (2024). A Novel Approach of Detecting Malicious Phishing URL Using Self-organizing Map. *Automatic Control and Emerging Technologies*, 259-268. https://doi.org/10.1007/978-981-97-0126-1_23

Schoenfeld, J. (2024). Cyber risk and voluntary Service Organization Control (SOC) audits. *Review of Accounting Studies*, *29*(1), 580-620. https://doi.org/10.1007/s11142-022-09713-0

Shi, W., & Liu, X. (2022). Research on SQL Injection Defense Technology Based on Deep Learning. *Artificial Intelligence and Security*, 538-549. https://doi.org/10.1007/978-3-031-06788-4_45

Siriwardena, P. (2019). *Base64 URL Encoding | SpringerLink*. https://springerlink.proxyusc.elogim.com/chapter/10.1007/978-1-4842-2050-4_20

Thexssrat. (2023, febrero 9). The popping history of XSS. *Medium*. https://thexssrat.medium.com/the-popping-history-of-xss-4122e34ac586

Trautman, L. J., & Ormerod, P. C. (s. f.). CORPORATE DIRECTORS' AND OFFICERS' CYBERSECURITY STANDARD OF CARE: THE YAHOO DATA BREACH. *AMERICAN UNIVERSITY LAW REVIEW*, *66*.

Viegas, V., & Kuyucu, O. (2022). Security Testing and Attack Simulation Tools. En *IT Security Controls* (pp. 263-290). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7799-7_9

Xia, P., Guo, Y., Lin, Z., Wu, J., Duan, P., He, N., Wang, K., Liu, T., Yue, Y., Xu, G., & Wang, H. (2024). WalletRadar: Towards automating the detection of vulnerabilities in browser-based cryptocurrency wallets. *Automated Software Engineering*, *31*(1), 1-33. https://doi.org/10.1007/s10515-024-00430-3

Xu, T., Singh, K., & Rajivan, P. (2023). Personalized persuasion: Quantifying susceptibility to information exploitation in spear-phishing attacks. *Applied Ergonomics*, *108*, 103908. https://doi.org/10.1016/j.apergo.2022.103908