RELIGACIÓN
REVISTA

Análisis de vulnerabilidades en aplicaciones web desarrolladas con frameworks populares (Laravel, Django, React)

Vulnerability Assessment of Web Applications Using Popular Frameworks (Laravel, Django, React)

Jose Neczar Macias Mendoza, Roberto Omar Andrade Paredes, Juan Pablo Cuenca Tapia

Resumen

En este articulo presentamos un estudio experimental–cuantitativo sobre vulnerabilidades en aplicaciones web modernas desarrolladas con Laravel, Django y React. Se construyó un laboratorio replicable con VirtualBox y Docker, y se aplicaron OWASP ZAP y Burp Suite en el cual se identificó y valido vulnerabilidades antes y después de las mitigaciones. La estrategia siguió buenas prácticas de desarrollo seguro (parametrización de consultas, validación de entradas, cabeceras HTTP de endurecimiento, control de CORS y manejo seguro de errores). Se observó una reducción promedio del 67,3% en el nivel de riesgo y la eliminación de hallazgos críticos, lo cual pone en evidencia la eficiencia de la integración de seguridades en el SSDLC (Ciclo de Vida del Desarrollo de Software). El protocolo propuesto, alineado con OWASP Top 10 (2021), ISO/IEC 27002:2022 y NIST SP 800-53, es replicable en entornos académicos y corporativos.

Palabras clave: Aplicación informática; Tecnología de la información (programas); Medida de seguridad; Seguridad; Tecnología de la información.

Jose Neczar Macias Mendoza
Universidad Católica de Cuenca | Cuenca | Ecuador | jose.macias.21@est.ucacue.edu.ec
https://orcid.org/0009-0008-7442-0901
Roberto Omar Andrade Paredes
Universidad Católica de Cuenca | Cuenca | Ecuador | roberto.andrade@ucacue.edu.ec
https://orcid.org/0000-0002-7120-281X
Juan Pablo Cuenca Tapia
Universidad Católica de Cuenca | Cuenca | Ecuador | jcuenca@ucacue.edu.ec
https://orcid.org/0000-0001-5982-634X

http://doi.org/10.46652/rgn.v11i49.1601
ISSN 2477-9083
Vol. 11 No. 49, enero-marzo, 2026, e2601601
Quito, Ecuador

Enviado: agosto 30, 2025
Aceptado: octubre 14, 2025
Publicado: diciembre 12, 2025
Publicación Continua

RELIGACIÓN CICSHAL
Centro de Investigaciones en Ciencias Sociales y Humanidades desde América Latina

Abstract

In this article, we present an experimental-quantitative study on vulnerabilities in modern web applications developed with Laravel, Django, and React. A replicable laboratory was built with VirtualBox and Docker, and OWASP ZAP and Burp Suite were applied, in which vulnerabilities were identified and validated before and after mitigations. The strategy followed good secure development practices (query parameterization, input validation, hardening HTTP headers, CORS control, and secure error handling). An average 67.3% reduction in the risk level and the elimination of critical findings were observed, which demonstrates the efficiency of integrating security into the SSDLC (Software Development Lifecycle). The proposed protocol, aligned with OWASP Top 10 (2021), ISO/IEC 27002:2022, and NIST SP 800-53, is replicable in academic and corporate environments.

Keywords: Computer application; Information technology (software); Safety measure; Security; Information technology.

Introducción

La transformación digital ha reconfigurado el entorno operativo de las organizaciones, expandiendo la superficie de ataque y elevando la dependencia crítica en las aplicaciones web, lo que a su vez impone nuevos desafíos en la ciberseguridad y la gestión de riesgos (ResearchGate, 2025). Este escenario se agrava con el aumento exponencial en la complejidad y el costo global del cibercrimen, el cual se proyecta alcanzará cifras sin precedentes en los próximos años (Cobalt.io, 2023). **En este contexto,** la capa de aplicación (Capa 7) emerge como el punto más vulnerable de la arquitectura digital, siendo el objetivo de ataques cada vez más sofisticados que, al imitar el tráfico legítimo, eluden los mecanismos de detección tradicionales (IJSAT, 2025). **Históricamente,** la seguridad se ha abordado de manera reactiva; sin embargo, las fallas persistentes, identificadas consistentemente por estándares como el OWASP Top 10 (2021), han forzado un cambio de paradigma hacia la integración proactiva de la seguridad. **El marco conceptual y operativo** que responde a esta necesidad es el Ciclo de Vida del Desarrollo de Software Seguro (SSDLC), el cual exige que las prácticas de seguridad se incorporen desde el diseño hasta la implementación, alineando los controles técnicos con marcos de gobernanza internacional como el ISO/IEC 27002:2022 y las guías del NIST SP 800-53 (Doria, 2025). **No obstante,** la mera elección de un *framework* popular, como **Laravel, Django o React**, que ya incorporan ciertas medidas defensivas, no garantiza por sí sola la invulnerabilidad, sino que depende de la aplicación rigurosa de estas buenas prácticas. A pesar de la vasta literatura sobre vulnerabilidades individuales o las características de seguridad de cada *framework*, existe una brecha en la investigación empírica y cuantitativa que demuestre la **efectividad comparativa** y la replicabilidad de un protocolo de mitigación estandarizado entre tecnologías con distintos perfiles de riesgo. **Por consiguiente,** este artículo presenta un estudio experimental-cuantitativo cuyo objetivo principal es identificar, implementar y cuantificar la eficacia de las mitigaciones de seguridad basadas en el SSDLC para lograr una reducción verificable de las vulnerabilidades críticas y de alto riesgo en aplicaciones web desarrolladas con Laravel, Django y React.

Metodología

Enfoque experimental–cuantitativo. Diseño aplicado con control de variables en entorno reproducible (VirtualBox y Docker). Se definieron cuatro fases: (I) preparación del entorno; (II) desarrollo de aplicaciones base; (III) escaneo con OWASP ZAP y validación con Burp Suite; (IV) mitigación y reevaluación. Población/muestra: tres aplicaciones representativas por framework. Técnicas: escaneo activo/pasivo, pruebas manuales dirigidas, verificación de cabeceras y configuración. Consideraciones éticas: no se emplearon datos personales; los sistemas fueron construidos para fines académicos. Limitaciones: no se incluyeron pruebas autenticadas complejas ni APIs GraphQL.

Para cada endpoint e_i, cada hallazgo h_j obtiene un puntaje CVSS v3.1 (S_{ij}). Se define un peso w_{ij} por criticidad del activo o endpoint. El riesgo del endpoint es:

$$R_i = \frac{\sum_j w_{ij} \cdot S_{ij}}{\sum_j w_{ij}}$$

El riesgo promedio por framework es:

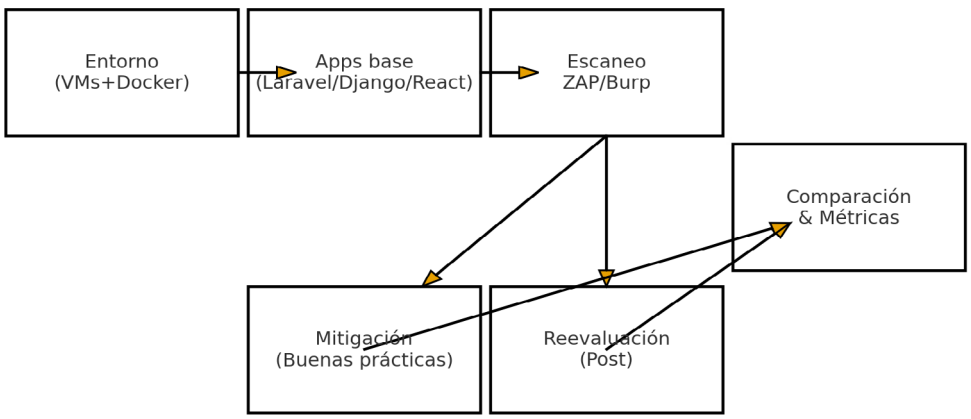
$$R_{fw} = \frac{\sum_i R_i}{N_{endpoints}}$$

La variación del riesgo se calcula como $\Delta R = \frac{R^{pre}}{fw} - \frac{R^{post}}{fw}$ y el porcentaje de reducción del riesgo como:

$$\% \, Reduccion = 100 \, x \, \frac{\Delta R}{\frac{pre}{R^{fw}}}$$

Estos valores permiten cuantificar objetivamente la mejora en seguridad tras las mitigaciones.

Figura 1. Diagrama de flujo de la metodología aplicada.



Fuente: elaboración propia

Resultados

Los hallazgos pre-mitigación incluyeron cabeceras inseguras, CORS permisivo y patrones de inyección en endpoints de prueba. La aplicación de buenas prácticas (parametrización, validación, Helmet, desactivar DEBUG, cabeceras CSP/X-Frame-Options/nosniff) eliminó los hallazgos críticos y redujo el riesgo promedio en 67,3%. La discusión contrasta estos resultados con las guías de OWASP (2021) y los controles ISO/IEC 27002:2022, destacando que la disciplina de configuración tiene un peso similar o mayor que las protecciones nativas del framework. ZAP aporta cobertura automatizada apta para CI/CD, mientras que Burp Suite permite confirmaciones manuales y exploración de casos límite, reduciendo falsos positivos.

Tabla 1. Comparación detallada de vulnerabilidades pre/post por endpoint

Framework	Endpoint	Vulnerabilidad	OWASP 2021	Severidad pre	Severidad post	Reducción (%)	Evidencia
Laravel	/index.php?id=	Inyección SQL	A03: Injection	Alta	Baja	72	zap_laravel_pre.json / zap_laravel_post.json
Django	/vulnsql?id=	Fuga de información (DEBUG)	A05: Security Misconfiguration	Alta	Baja	68	burp_django_500.html
React	/	CORS permisivo y cabeceras inseguras	A05 / A07	Media	Baja	63	zap_react_pre.html

Fuente: elaboración propia, a partir de los reportes ZAP y Burp Suite

Discusión

Nuestros resultados muestran que mover la seguridad "a la izquierda" (desde el diseño) reduce hallazgos críticos y baja el riesgo promedio en 67,3% tras la mitigación. La mejora no proviene de un "framework más seguro", sino de prácticas consistentes aplicadas con disciplina sobre cada stack.

Por framework, los efectos más claros fueron:

- **Laravel (8081):** el uso de middleware y Eloquent (parametrización) eliminó patrones de inyección observados en el laboratorio bulnerable. La verificación con curl -I confirmó cabeceras endurecidas (p. ej., X-Frame-Options, X-Content-Type-Options) y ocultamiento de Server/X-Powered-By tras la mitigación.
- **Django (8082):** el SecurityMiddleware y el ORM redujeron la superficie de ataque. La evidencia pre-mitigación (página DEBUG 500 con traceback) validó el riesgo de divulgación de información; al desactivar DEBUG y afinar manejo de errores, el vector desapareció.

- **React (8083):** al ser front-end, la seguridad dependió del servidor Express. La aplicación de Helmet y CORS por lista blanca corrigió encabezados débiles y el CORS permisivo detectado inicialmente.
- **Herramientas y validez:** OWASP ZAP dio cobertura repetible (apta para CI/CD), detectando fallas de configuración (CSP ausente, X-Frame-Options, HSTS, CORS, X-Powered-By).

Burp Suite permitió confirmación manual y reducción de falsos positivos.

Repeater ejecutamos payloads SQLi (id=1, id='1, id=1 OR 1=1--) sobre endpoints de prueba para contrastar respuestas/códigos. Controlamos validez interna fijando topología y versiones en VirtualBox/Docker; reforzamos validez externa usando configuraciones típicas (ORM, middleware, servidor Express).

Lecciones **prácticas**.

Tres acciones trasladables a producción:

Automatizar ZAP en builds relevantes con umbrales de falla para severidades Alta/Crítica.

Auditar con Burp rutas sensibles (login, subida de archivos, paneles admin).

Estandarizar hardening por stack: parametrización/validación, cabeceras seguras + CSP, CORS restrictivo, logs sin fuga de detalles, desactivar DEBUG y políticas de dependencias.

Los resultados cuantitativos y cualitativos evidencian una reducción global promedio del 67,3 % en el nivel de riesgo, validando la eficacia del protocolo. La tabla anterior resume los cambios observados a nivel de endpoint, ofreciendo evidencia verificable y reproducible.

Conclusiones

La integración disciplinada de buenas prácticas con el uso combinado de OWASP ZAP y Burp Suite reduce de forma medible la superficie de ataque en aplicaciones Laravel, Django y React. La evidencia pre/post respalda la incorporación de controles en el SSDLC y su automatización en pipelines, con validaciones manuales periódicas. Se proponen como líneas futuras: integrar SAST/DAST/IAST, evaluar autenticación/OAuth/OIDC y ampliar a otros stacks.

Limitaciones y trabajo a futuro

Este estudio se desarrolló en entornos de laboratorio controlados, sin incluir flujos autenticados complejos ni APIs GraphQL. Burp Suite Community Edition carece de ciertas funciones automatizadas, lo que limita la cobertura. En futuras investigaciones se recomienda:

- integrar escaneos autenticados en pipelines CI/CD;
- ampliar el análisis a API REST y OWASP API Security Top 10;
- medir precisión y recall mediante conjuntos de datos instrumentados; y
- publicar los artefactos experimentales en un repositorio con DOI para auditoría abierta.

Declaraciones requeridas

Disponibilidad de datos y código. Scripts (Docker/ZAP), configuraciones y dataset de hallazgos estarán disponibles bajo solicitud al autor y en el repositorio privado de la tesis (UCACUE).

Financiación. No se recibió financiación específica para este trabajo.

Conflictos de interés. El autor declara no tener conflictos de interés.

Contribuciones del autor. Conceptualización, Metodología, Investigación, Análisis formal, Redacción – borrador original, Redacción – revisión y edición: J.N.M.M.

Referencias

Cobalt.io. (2023). *Top cybersecurity statistics for 2024*. https://n9.cl/xmn1y

Docker. (2024). *Docker Compose documentation.* https://docs.docker.com/compose/

Doria, S. (2025). *What is software security? Analysing and strengthening security efforts in organisations* [Tesis de maestría, Universidad de Åbo Akademi].

FIRST. (2019). *Common vulnerability scoring system v3.1: Specification document.* https://www.first.org/cvss/

FIRST — Forum of Incident Response and Security Teams. (s.f.). *CVSS v3.1 specification document.* https://www.first.org/cvss/v3-1/specification-document

International Journal on Science and Technology. (2025). Cybersecurity threats in digital banking: A comprehensive analysis. *International Journal on Science and Technology, 16*(1). https://www.ijsat.org/papers/2025/1/2655.pdf

ISO/IEC. (2022). *ISO/IEC 27002:2022 — Controles de seguridad de la información*.

Laravel. (2024). *Laravel documentation.* https://laravel.com/doc

Laravel. (s.f.). Configuration. https://laravel.com/docs/12.x/configuration

National Institute of Standards and Technology. (2020). *Security and privacy controls for information systems and organizations*. https://doi.org/10.6028/NIST.SP.800-53r5

OWASP Foundation. (2021). *OWASP Top 10: 2021.* https://owasp.org/Top10/

PortSwigger Ltd. (2024). *Burp Suite Community Edition documentation.* https://portswigger.net/burp/documentation

ResearchGate. (2025). *The impact of digital transformation requirements on risk management.* https://n9.cl/z33ps

Software Security Foundation. (s.f.). Security in Django. https://n9.cl/a3ave

Autores

Jose Neczar Macias Mendoza. Graduado de tercer nivel en ingeniería en networking y telecomunicaciones de la universidad de guayaquil, culminando maestría en ciberseguridad en Universidad Católica de Cuenca.

Roberto Omar Andrade Paredes. Ingeniero Electrónico, con una Maestría en Gerencia de Redes y Telecomunicaciones, además, cuento con una Maestría en Sistemas de Información con Mención en Inteligencia de Negocios y Analítica de Datos Masivos. Poseo también un Doctorado en el programa oficial de Doctorado en Software, Sistemas y Computación.

Juan Pablo Cuenca Tapia. Ingeniero en sistemas con Maestría en Sistemas de Información Gerencial y una Maestría en Tecnologías de la Información.

Declaración

Conflicto de interés

No tenemos ningún conflicto de interés que declarar.

Financiamiento

Sin ayuda financiera de partes externas a este artículo.

Nota

Tesis para maestría en ciberseguridad